

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-024293

(43)Date of publication of application : 25.01.2002

(51)Int.Cl.

G06F 17/30
H04L 12/56

(21)Application number : 2001-104687

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>

(22)Date of filing : 03.04.2001

(72)Inventor : CALVIGNAC JEAN LOUIS
HEDDES MARCO C
JEFFERIES CLARK DEBUS
PATEL PIYUSH CHUNILAL
RINALDI MARK ANTHONY

(30)Priority

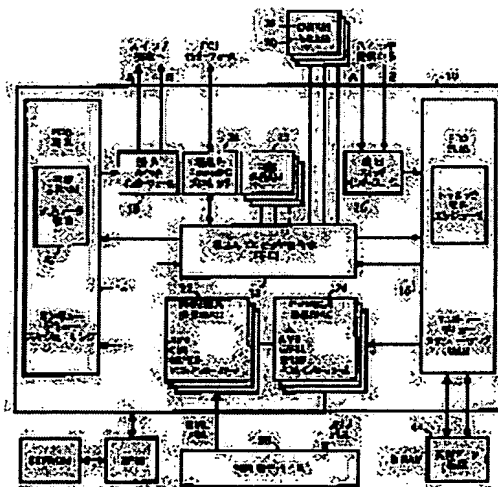
Priority number : 2000 545100 Priority date : 06.04.2000 Priority country : US

(54) PATTERN RANGE COMPARISON IMPLEMENTING METHOD, COMPUTER-READABLE MEDIUM, AND DEVICE

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a new data structure, a method, and a device for a software management tree(SMT) which provide a control point processor.

SOLUTION: This retrieving mechanism reduces the storage space of a node by using only a forward pointer together with a next bit or bit group to be tested next. Multiple retrieval is unnecessary and filter rules for an application are processed to make it possible to connect various filter rules. Two patterns of the same length are stored in respective leaves to define range comparison. Final comparing operation is comparison within a range or comparison below a mask. Through the comparison within the range, it is decided whether or not an input key is within the range defined by the two patterns. Through the comparison below the mask, various bits in the input key are compared with various bits in a 1st leaf pattern under the mask specified with the 2nd leaf pattern.



LEGAL STATUS

[Date of request for examination]

03.04.2001

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開2002-24293

(P2002-24293A)

(43)公開日 平成14年1月25日(2002.1.25)

(51)Int.Cl. ⁷	識別記号	F I	テマコード*(参考)
G 0 6 F 17/30	4 1 2	G 0 6 F 17/30	4 1 2 5 B 0 7 5
	1 1 0		1 1 0 C 5 K 0 3 0
	4 1 4		4 1 4 A
H 0 4 L 12/56	1 0 0	H 0 4 L 12/56	1 0 0 Z

審査請求 有 請求項の数42 OL (全 26 頁)

(21)出願番号 特願2001-104687(P2001-104687)
 (22)出願日 平成13年4月3日(2001.4.3)
 (31)優先権主張番号 09/545100
 (32)優先日 平成12年4月6日(2000.4.6)
 (33)優先権主張国 米国(US)

(71)出願人 390009531
 インターナショナル・ビジネス・マシー
 ズ・コーポレーション
 INTERNATIONAL BUSIN
 ESS MACHINES CORPO
 RATION
 アメリカ合衆国10504、ニューヨーク州
 アーモンク (番地なし)
 (74)代理人 100086243
 弁理士 坂口 博 (外2名)

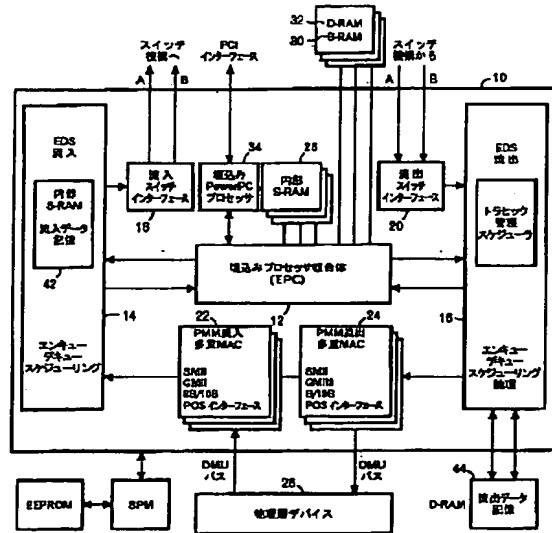
最終頁に続く

(54)【発明の名称】 パターン範囲比較実行方法、コンピュータ可読媒体および装置

(57)【要約】 (修正有)

【課題】 制御ポイント・プロセッサを提供するソフト
 ウェア管理ツリー(SMT)のための新奇なデータ構
 造、方法および装置を提供すること。

【解決手段】 本検索機構は、テストすべき次のビット
 またはビット・グループとともに前方ポイントのみを使用
 し、ノードの記憶スペースを減少させる。また複数の
 検索は不要で、アプリケーションに対する複数のフィル
 タ・ルールを処理し、様々なフィルタ・ルールを結びつ
 けることも可能にする。同じ長さの2つのパターンが、
 それぞれのリーフに記憶されて範囲比較を定義する。最
 終比較動作は、範囲下の比較かマスク下の比較のいづれ
 かである。範囲下の比較では、入力キーが2つのパター
 ンによって定義された範囲内にあるかが判定される。マ
 スク下の比較では、入力キー内の諸ビットが、第2リー
 フ・パターンで特定されたマスク下で、第1リーフ・パ
 ターン内の諸ビットと比較される。



【特許請求の範囲】

【請求項1】ソフトウェア管理ツリー内の可変長検索キーに対して、コンピュータ処理デバイスによりパターン範囲比較を実行する方法であって、
検索文字列として入力キーを読み込む行為と、
検索キーの最上位Nビットを、検索ツリーの複数のルート・ノードを表すテーブルに対するインデックスとして使用する行為であって、非空のエントリのそれぞれが検索ツリー内の次の分岐またはリーフに対するポインタを含む行為と、

非空のテーブル・エントリ内の前記ポインタが、対応する検索ツリーのリーフまたは次の分岐のどちらを指すかを判定する行為と、

前記ポインタが前記対応する検索ツリーのリーフを指さない場合に、次の分岐内容を読み込む行為と、
対応する検索ツリーのリーフに到達したときにリーフ内容を読み込み、前記リーフ内の一対のパターンと前記検索キーを比較して、前記一対のリーフ・パターンによって定義された範囲が前記検索キーを含むかどうかを判定する行為と、

前記リーフ・パターンが前記検索キーを含む場合に、見つかったリーフの内容を、要求するアプリケーションに戻す行為とを含む方法。

【請求項2】検索キーを形成するためにプログラム可能ハッシュ関数を使用して入力キーをハッシュすることをさらに含む、請求項1に記載のパターン範囲比較を実行する方法。

【請求項3】検索ツリーの複数のルート・ノードを表す前記テーブルが2^N個のエントリを含む、請求項1に記載のパターン範囲比較を実行する方法。

【請求項4】前記コンピュータ処理デバイスがネットワーク・プロセッサである、請求項1に記載のパターン範囲比較を実行する方法。

【請求項5】前記対応する検索ツリーの前記次の分岐の内容が他の次の分岐を指す、請求項1に記載のパターン範囲比較を実行する方法。

【請求項6】前記次の分岐の内容が前記対応する検索ツリーの前記リーフを指す、請求項1に記載のパターン範囲比較を実行する方法。

【請求項7】前記リーフ・パターンが前記検索キーを含まず、他のリーフに対するポインタを含まない場合に、成功表示を戻さないことをさらに含む、請求項1に記載のパターン範囲比較を実行する方法。

【請求項8】前記テーブルに対する前記インデックスが空エントリを指している場合に、成功表示を戻さないことをさらに含む、請求項1に記載のパターン範囲比較を実行する方法。

【請求項9】前記検索キーにカラー・レジスタの内容を追加して最終検索キーを提供することをさらに含む、請求項1に記載のパターン範囲比較を実行する方法。

【請求項10】前記検索キーに0の文字列を追加して最終検索キーを提供することをさらに含む、請求項1に記載のパターン範囲比較を実行する方法。

【請求項11】前記一対のパターンによって定義される範囲内にあるかどうかを判定するためにチェックされる整数として前記検索キー内の諸ビットが扱われる、範囲下の比較動作を、一対のパターンを比較する前記行為が含む、請求項1に記載のパターン範囲比較を実行する方法。

10 【請求項12】第2リーフ・パターン内に指定されたマスク下で第1リーフ・パターン内の諸ビットと前記検索キー内の諸ビットが比較される、マスク下の比較動作を、一対のパターンを比較する前記行為が含む、請求項1に記載の方法。

【請求項13】前記リーフが他のリーフに対する連鎖ポインタを含む場合に、他のリーフ内に記憶された一対のパターンを読み込み、前記パターンと前記検索キーを比較する行為と、

記憶された前記パターンが前記検索キーを含まず、連鎖内の次のリーフに対するポインタを含まない場合に、成功表示を戻さない行為とをさらに含む、請求項1に記載のパターン範囲比較を実行する方法。

【請求項14】前記リーフが他のリーフに対する連鎖ポインタを含む場合に、他のリーフに記憶された一対のパターンを読み込み、前記パターンと前記検索キーを比較する行為と、

記憶された前記パターンが前記検索キーを含む場合に、成功表示を戻す行為とをさらに含む、請求項1に記載のパターン範囲比較を実行する方法。

30 【請求項15】ソフトウェア管理ツリー内の可変長検索キーに対してパターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体であって、
検索されるべきパターンまたはキーと、
検索ツリーに対する第1アドレス・ロケーションを記憶する直接テーブルと、

前記検索ツリー内の分岐をそれぞれ表す複数のパターン検索制御ブロックと、
各エントリに関連付けられた少なくとも1つの範囲比較を指定する比較テーブルと、

40 複数のリーフであって、前記検索キーと比較する一対のパターンを各リーフが記憶するリーフとを含むコンピュータ可読媒体。

【請求項16】ツリー検索メモリを管理するルックアップ定義テーブルをさらに含む、請求項15に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項17】前記ルックアップ定義テーブルが、前記ツリーが中にある物理メモリ、前記キーおよびリーフのサイズ、ならびに実行されるべき検索のタイプ、を定義するエントリを含む、請求項15に記載の、パターン範

囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項18】前記ルックアップ定義テーブルが、複数のメモリに実施される、請求項15に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項19】直接テーブル・エントリのフォーマットが、少なくとも1つの検索制御ブロック、次のパターン検索制御ブロックを指す次のパターン・アドレス、リーフまたは結果を指すリーフ制御ブロック・アドレス、テストすべき次のビットまたは諸ビット、および直接リーフを含む、請求項15に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項20】パターン検索制御ブロックのフォーマットが、少なくとも1つの検索制御ブロック、次のパターン検索制御ブロックを指す次のパターン・アドレス、リーフまたは結果を指すリーフ制御ブロック・アドレス、およびテストすべき次のビットまたは諸ビットを含む、請求項15に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項21】前記比較テーブルが、少なくとも1つの範囲比較を定義するエントリを含み、各範囲比較が、前記フィールドの第1ビットの位置であるオフセット・パラメータおよび前記フィールドのビット長である長さパラメータによって定義される、請求項15に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項22】リーフ・データ構造が、少なくとも1つのリーフ連鎖ポイント、プレフィクス長、前記検索キーと比較すべき一対のパターン、および可変ユーザ・データを含む、請求項15に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項23】前記直接リーフが、直接テーブル・エントリに直接記憶され、検索制御ブロック、および検索キーと比較すべき一対のパターンを含む、請求項15に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項24】パターン検索制御ブロックが、前記検索ツリー内のリーフ・パターンが変化する位置に挿入される、請求項15に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項25】パターン検索制御ブロックが、幅1および高さ1により定義される形を有し、少なくとも36ビットのライン長を有するメモリ内に記憶される、請求項15に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項26】ソフトウェア管理ツリー内の可変長検索キーに対してパターン範囲比較を実行するための、半導

体基板上に組立てられた装置であって、複数のプロトコル・プロセッサ、およびフレーム処理を提供する内部制御ポイント・プロセッサを含む組込みプロセッサ複合体と、各プロトコル・プロセッサにアクセス可能で、高速パターン検索、データ操作、およびフレーム分析を提供する複数のハードウェア・アクセラレータ・コプロセッサと、

少なくとも1つの検索ツリーを表す複数のデータ構造を記憶する複数のプログラム可能メモリ・デバイスであって、前記データ構造が、直接テーブル、パターン検索制御ブロック、比較テーブル、および前記検索キーと比較すべき一対のパターンを含むリーフを含むプログラム可能メモリ・デバイスと、

前記複数のメモリ・デバイスに対する各プロトコル・プロセッサのアクセスを制御する制御メモリ・アービタとを備える装置。

【請求項27】メモリ読みおよび書き込みならびにメモリ範囲チェックを含むツリー検索命令を実行するプロトコル・プロセッサの実行と並行して動作するツリー検索エンジンをさらに備える、請求項26に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項28】前記複数のメモリ・デバイスが、少なくとも1つの内部静的ランダム・アクセス・メモリ、外部静的ランダム・アクセス・メモリ、および外部動的ランダム・アクセス・メモリをさらに備える、請求項26に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項29】前記制御メモリ・アービタが、前記複数のプロトコル・プロセッサと前記複数のメモリ・デバイスの間でメモリ・サイクルを割り振ることによって制御メモリ動作を管理する、請求項26に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項30】各プロトコル・プロセッサが、プライマリ・データ・バッファ、スクラッチ・パッド・データ・バッファおよびデータ記憶動作のための制御レジスタを備える、請求項26に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項31】前記検索キーに対して幾何学的ハッシュ関数を実行するハッシュ・ボックス・コンポーネントをさらに備える、請求項26に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項32】プログラム可能検索キー・レジスタおよびプログラム可能ハッシュ化キー・レジスタをさらに備える、請求項26に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項33】複数の独立した検索ツリー間で単一のテーブル・データ構造を共用することを可能にするための

プログラム可能カラー・キー・レジスタをさらに備える、請求項32に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項34】前記カラー・レジスタの内容が、もしイネーブルであれば、前記ハッシュ出力に追加されて最終ハッシュ化キーを生成する、請求項33に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項35】前記カラー・レジスタがイネーブルでない場合に、最終ハッシュ化キー生成のために前記ハッシュ出力に同数の0を追加する、請求項33に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項36】ソフトウェア管理ツリー内の可変長検索キーに対してパターン範囲比較を実行するコンピュータ・プログラム・プロダクトを含むコンピュータ可読媒体であって、

入力キーを検索文字列として読込むプログラム命令と、前記検索キーの最上位Nビットを、検索ツリーの複数のルート・ノードを表すテーブルに対するインデックスとして使用するプログラム命令であって、それぞれの非空のエントリが、前記検索ツリー内の次の分岐またはリーフに対するポインタを含むプログラム命令と、非空のテーブル・エントリ内の前記ポインタが、前記対応する検索ツリーのリーフまたは次の分岐のどちらを指すかを判定するプログラム命令と、前記ポインタが、前記対応する検索ツリーの前記リーフを指さない場合に、前記次の分岐内容を読込むプログラム命令と、対応する検索ツリーの前記リーフに到達したときに前記リーフ内容を読み込み、前記リーフ内の一対のパターンと前記検索キーを比較して前記一対のリーフ・パターンによって定義された範囲が前記検索キーを含むかどうかを判定するプログラム命令と、前記リーフ・パターンが前記検索キーを含む場合に、見つかった前記リーフの内容を前記要求するアプリケーションに戻すプログラム命令とを含むコンピュータ可読媒体。

【請求項37】前記テーブルに対する前記インデックスが空のエントリを指している場合に、成功表示を戻さないプログラム命令をさらに含む、請求項36に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

【請求項38】カラー・レジスタの内容を前記検索キーに追加して最終検索キーを提供するプログラム命令をさらに含む、請求項36に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

【請求項39】前記検索キーに0の文字列を追加して最終検索キーを提供するプログラム命令をさらに含む、請求項36に記載の、パターン範囲比較を実行するコンピ

ュータ・プログラム・プロダクト。

【請求項40】前記検索キー内の前記諸ビットが、前記一対のパターンによって定義される範囲内にあるかどうかを判定するためにチェックされる整数として扱われる、範囲下の比較動作を実行するプログラム命令をさらに含む、請求項36に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

【請求項41】前記検索キー内の諸ビットが、第1リーフ位置の諸ビットと、第2リーフ・パターンに指定されたマスク下で比較される、マスク下の比較動作を実行するプログラム命令をさらに含む、請求項36に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

【請求項42】前記リーフが他のリーフに対する連鎖ポインタを含む場合に、他のリーフに記憶された一対のパターンを読み込み、前記パターンと前記検索キーを比較するプログラム命令と、

記憶された前記パターンが、前記検索キーを含まず、前記連鎖内の次のリーフに対するポインタを含まない場合に、成功表示を戻さないプログラム命令とをさらに含む、請求項36に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、一般にパターン・マッチング・アルゴリズムに関し、より詳細には、ネットワーク処理デバイス内に実施することのできるソフトウェア管理ツリー検索アルゴリズムに関する。

【0002】関連出願の相互参照

本出願は、同時係属で、共通の譲渡人に譲渡された、米国特許出願番号09/384691で1999年8月27日出願の「Network Processor Processing Complex and Methods」という表題の特許出願と、同じく出願番号09/54353の「Full Match (FM) Search Algorithm Implementation for a Network Processor」という表題の特許出願と、同じ出願番号09/544992の「Longest Prefix Match (LPM) Algorithm Implementation for a Network Processor」という表題の特許出願とに関係し、必要により参照されたい。

【0003】

【従来の技術】より一層複雑なタスクをメディア・スプードでサポートする、ハードウェアによって統合された処理に対する要求が、ネットワーク・プロセッサの創造につながった。ネットワーク・プロセッサは、1組の組み込まれた、プログラム可能プロトコル・プロセッサおよび補足的システム・コプロセッサを介して、機能柔軟性を有するワイヤスピード・フレーム処理/転送機能を提供する。ネットワーク・プロセッサは、マイクロプロセッサが今日のパーソナル・コンピュータに対するのと同様に、ネットワークに対する基本的なネットワーク構

成要素になると期待されている。ネットワーク・プロセッサは、複数のデータ・ストリームのリアルタイム処理、高度なセキュリティの提供、ならびにIPパケット処理／転送機能を提供する。さらにこのネットワーク・プロセッサは、並列分散処理およびパイプライン処理設計などの高度なアーキテクチャを介してスピードの改善を可能にする。これらの機能は、効率的検索エンジン、データ処理スループット向上を可能にし、複雑なタスクの迅速な実行を可能にする。ネットワーク・プロセッサのプログラム可能な特徴は、新たなカスタムASIC (Application Specific Integrated Circuit: 特定用途向けIC) 設計を必要とすることなく、新たなプロトコルおよび技術を実施するためのより容易な移行方法を、ネットワーク・プロダクト開発者に提供する。

【0004】ネットワーク・プロセッサは、インターネットまたは企業ネットワークのプロバイダのための相互接続ソリューションを開発するために、高度にカスタマイズ可能でスケラブルな技術を提供する。ネットワーク・プロセッサは、ローエンドのスタンドアロン・デバイスから大きなマルチラック・ソリューションまで、広範囲のソリューションのための基礎を提供する。この種のスケリングは、高性能、非ブロック化パケット・ルーティング・スイッチ技術、および他の業界スイッチ技術に適合することのできる、IBM CorporationのDASL (Data Aligned Serial Link) インタフェースなどの独占インタフェースの使用により達成することができ

る。

【0005】プログラム可能通信集積回路としてネットワーク・プロセッサは、非常に効率的なパケット分類、フレーム毎のマルチテーブル・ルックアップ、パケット修正、待ち行列／ポリシ管理、その他のパケット処理機能を提供する。ネットワーク・プロセッサは、スイッチング・エンジン、検索エンジン、フレーム・プロセッサおよびEthernet MACを1つのデバイス上に統合して、大容量メディア・ウェイト・スイッチング・フレームを必要とする顧客のニーズを任意のプロトコル層のフレーム内容に基づいてサポートする。

【0006】ハードウェア・アクセラレータは、フレーム転送、フレーム・フィルタリングおよびフレーム変更を実行する。複雑な範囲およびアクション仕様を有する数百のルールを執行するネットワーク・プロセッサの能力は、フィルタリング機能に対する新たなベンチマークを設定し、ネットワーク・プロセッサに基づくシステムを大容量サーバ・ファーム・アプリケーションに一意に適合させる。

【0007】ネットワーク・プロセッサと共に開発される典型的システムは、分散ソフトウェア・モデルを用い、それぞれのプログラム可能ネットワーク・プロセッサは諸タスクを同時に実行する。いくつかの機能は、制御ポイント(CP)プロセッサ内で実行され、このプロ

セッサはネットワーク・プロセッサの内部または外部でよい。CPは、層2および層3のルーティング・プロトコル、ならびに層4および層5のネットワーク・アプリケーションおよびシステム管理に対するサポートを提供する。ワイヤスピード転送／フィルタリング機能は、ネットワーク・プロセッサ・ハードウェアと常駐ビコードの組合わせによって実行される。

【0008】いくつかの相互接続されたノードを備える通信ネットワークにおいて、データは1つのノードから任意の他のノードまたはネットワークに送ることができる。ルータと呼ばれる特別なノードは、データをその宛先に転送することを担当する。通信ネットワークを介して送られる任意のデータは、宛先アドレスに関する情報を、一般にヘッダの一部として含む。各ルータは、この情報または少なくともその一部を、内部に記憶されたアドレスのリストと比較する。記憶されたアドレスと宛先アドレスの間に一致が見つかる場合、ルータは宛先ノードに至るパスを確立する。ネットワーク・サイズおよび構造次第で、データはその宛先に直接転送されるか、または他の中間ルータに送られる。国際標準化機構(ISO)は、ルータが部分アドレスに対するルーティング情報を記憶するルーティング標準を公表した。次いでルータは、そのルータが自分のデータベース内に有する、最も一致する部分アドレスにパケットを送る。ISO標準は、所与の桁数または所与のヘッダ長を用いて、階層構造のノードを構築することを可能にする。メイン・ルータはそのアドレスの最初の部分によってアドレス指定され、サブルータは中間の部分、最終の宛先はアドレスの最後の桁によってアドレス指定される。したがってどのルータも、データが送られる階層レベルに割り当てられた桁を読めば充分である。

【0009】受け取ったパケットのルーティングは、付随するアドレス文字列に基づく。アドレス文字列はデータベース内の検索キーとして用いられ、このデータベースは、アドレス文字列、ならびにパケット配信の際にどのルータが次であるかなど、他の適切な詳細を含む。データベースは、ルーティング・テーブルと呼ばれるが、現在のルータと次のルータの間のリンクは、パケットの進行中の次のホップと呼ばれる。ルーティング・テーブル検索処理は、アドレスの構造およびテーブルの構成に依存する。例えば、8ビットより小さいサイズで非階層構造を有する検索キーであれば、一連のアドレス・エントリとして構成されたルーティング・テーブル内で最も効率的に見つかる。検索キーは、正しいエントリを見つけるための、テーブル内のインデックスとして用いられる。より大きいサイズ、例えば32ビットの検索キーに対して、対応するルーティング・テーブルは10,000より多いエントリを有する可能性がある。インデックスによって直接検索される単純なテーブルとしてデータベースを構成すれば、テーブルのほとんどが空なので、

大量のメモリ・スペースを無駄にする。

【0010】従来型のルータは、検索処理をいくつかのステップに分解する。第1ステップは、ルータが宛先ホスト・コンピュータに直接接続されているかどうかを決定することである。この場合、メッセージは宛先から1ホップであり、その方向にルーティングされるべきである。宛先コンピュータがルータに直接接続されていない場合、次のステップは宛先ネットワークのトポロジ的方向を決定することである。トポロジ的レイアウトから方向が決定された場合、メッセージはその進路にルーティングされる。そうでない場合、最終ステップはデフォルト・リンクに沿ってメッセージをルーティングすることである。

【0011】通常第1ステップは、ルータに直接接続されたホスト・コンピュータの32ビット・アドレスを含むテーブル全体の直線検索を用いて実行される。局所的トポロジを反映して、アドレス・テーブル内の各エントリは、アドレス指定されたコンピュータに直接つながる対応する出力インタフェースに接続される。宛先アドレスがルータに受け取られると、32ビット全体がテーブル内の宛先アドレスのそれぞれと比較される。一致が見つかる場合、メッセージは、指定されたルータ・インタフェースを介して対応する宛先に直接送られる。

【0012】第2ステップ、すなわち宛先ネットワークの方向を決定するステップは、ネットワーク・アドレス数のためにそのようなテーブルを管理し使用するのが困難になるので、普通はテーブル全体の直線検索によって実行されない。従来技術においてアドレス文字列が3レベル階層のネットワーク・アドレス、サブネット・アドレス、およびホスト識別に準拠したとき、ルータは、ハッシング、パトリシア・ツリー検索、およびマルチレベル検索など、いくつかのよく知られている手法の1つを用いて決定を実行した。ハッシングにおいて、ハッシュ関数はアドレスのネットワーク部分を少なくし、小さい管理可能なインデックスを生成する。ハッシュ・インデックスは、ハッシュ・テーブルに索引付けすること、および一致するハッシュ・エントリを検索することのために用いられる。ハッシュ・テーブルの各ハッシュ・エントリには、対応するネットワークのトポロジ的方向を指す出力インタフェースのアドレスが対応している。ハッシュ・ネットワーク部分とハッシュ・エントリの間の一致が見つかる場合、メッセージは対応するインタフェースおよび宛先ネットワークに向けられる。

【0013】ハッシングは、大きい管理不能なフィールドを小さい管理可能なインデックスに減少させる。しかしこの処理において、2つ以上のフィールドが同じハッシュ・インデックスを生成する可能性がある。これらのフィールドはハッシュ・テーブル内の同じ位置に記憶されなければならないので、この事象は衝突と呼ばれる。

衝突中のエントリを区別するためには追加の検索が必要

である。したがって衝突は、ハッシング検索を用いることから得られる効率を減少させ、すべての許されるアドレスが単一のインデックスに減少する最悪の場合、ハッシングは検索処理として事実上無益になる。

【0014】パトリシア・ツリー検索は、ハッシング法が直面した衝突を回避する。この検索方法では、すべてのアドレス文字列、および関連するルート情報などの付随情報がバイナリ・ツリー内に記憶される必要がある。アドレス文字列中の最上位ビット位置から開始して、この検索処理はアドレスとツリー・ノードをビット毎に比較する。一致したビット値により、この検索が左または右のどちらの子ノードに行くかが導かれ、アドレスの次のビットに対して処理が繰り返される。検索時間は、記憶された最長のアドレス文字列のサイズに比例する。パトリシア・ツリー検索では、平均検索時間と最悪の場合の検索時間の間の相違はそれほど大きくない。さらに、ルーティング・テーブルは非常に効率的に構成される。この検索は、ハッシング法の同等のルーティング・テーブルより小さいメモリしか必要としない。パトリシア・ツリー検索は、最悪の場合の検索をハッシング法よりうまく処理するが、ほとんどの場合、一致を見つけるのにかなりより長くなる。したがって多数の従来型ルータは、ハッシングおよびパトリシア・ツリー検索の組合わせを用いる。この組合わせは、マルチレベル検索と呼ばれる。

【0015】マルチレベル検索は、ハッシングをパトリシア・ツリー検索と結びつける。キャッシュは、直近に、かつおそらく最も共通にルーティングされるネットワーク・アドレスのサブセットを含むハッシュ・テーブルを記憶するが、パトリシア・ツリーは、ネットワーク・アドレスの完全なセットを記憶する。メッセージが受け取られると、宛先アドレスがテーブル上にハッシングされる。所定の時間以内にその宛先アドレスが見つからない場合、もし記憶されていればアドレスが見つかることを保証するパトリシア・ツリー検索エンジンに、このアドレスが渡される。

【0016】従来技術には、固定一致ツリー、最長プレフィクス一致ツリー、およびソフトウェア管理ツリーを含む、いくつかの知られているツリー検索アルゴリズムがある。固定一致ツリーは、層2のEthernet MACテーブルなど、正確な一致を必要とする固定サイズのパターンに対して用いられる。最長プレフィクス一致ツリーは、IPサブネット転送など、部分的に一致しか必要としない可変長パターンに対して用いられる。ソフトウェア管理ツリーは、フィルタ・ルールなど、範囲またはビット・マスクと定義されるパターンに対して用いられる。一般にルックアップは、ツリー検索エンジン(TSE)の助けとともに実行される。

【0017】

【発明が解決しようとする課題】本発明の一目的は、複

11

数の検索を必要とすることなく、アプリケーションに対する複数のフィルタ・ルールを処理するユニークで効率的な機構を提供することである。

【0018】本発明の他の目的は、後方ポインタに対する記憶を必要とせず、テストすべき次のビットまたはビット・グループとともに前方ポインタだけしか使用しない検索機構を提供し、それによって、ノードに対する記憶スペースを減少させることである。

【0019】

【課題を解決するための手段】本発明は、制御ポイントによって定義される検索機構に従うツリー構造を作り出す機構を提供するソフトウェア管理ツリー (Software Managed Tree: SMT) のための新奇なデータ構造を述べる。この一例示的機構は、IPソース・アドレス (IPSA)、IP宛先アドレス (IPDA)、ソース・ポート (SP)、宛先ポート (DP)、および通信プロトコルを含むインターネット・プロトコル (IP) 5 タプル・フィルタリング・テーブルである。完全一致または最長プレフィクス一致ツリーとは対照的に、SMTツリーは範囲比較に対するサポートを可能にする。例えば、20 ソース・ポートがxとyの範囲内になければならないことを指定するためにリーフを用いることができる。この手法により、効率的な記憶/検索時間を有する非常に効率的で単純な実施が可能になる。この手法により、様々なフィルタ・ルールを結びつけることもまた可能になる。

【0020】

【発明の実施の形態】本発明が組み込まれるネットワーク・プロセッサの文脈において本発明を述べる。ネットワーク・プロセッサ10は、単一チップ上のプログラム可能スイッチングおよびルーティング・システムであり、そのアーキテクチャは図1に示されている。このネットワーク・プロセッサは、10/100 Ethernet、Gigabit Ethernet、およびPacket Over SONET (POS) 用のメディア・インタフェース、ならびにスイッチ・インタフェースに対する接続用のデータ配列シリアル・リンク (data aligned serial links: DASL) を提供する。内部ハードウェア・アクセラレータは、性能および効率を向上する。組込みプロセッサ複合体 (EPC) 12は、プロトコル・プロセッサ、ならびにフレーム処理、構成および管理サポートのための内部制御ポイント・プロセッサを含む。

【0021】最大N個の並列プロトコル・プロセッサが利用可能である。16プロトコル・プロセッサの実施形態では、16、384ワードの内部ピココード命令記憶および32、768ワードの外部ピココード命令記憶が利用可能であり、2、128MIPS (million instructions per second: 百万命令/秒) の合計処理能力を提供する。さらに各プロトコル・プロセッサは、高速パターンの検索、データ操作、内部チップ管理機能、フレー

12

ム分析、およびデータ・プリフェッチング・サポートを提供するM個のハードウェア・アクセラレータ・コプロセッサに対するアクセスを有する。好ましい実施形態においてプロトコル・プロセッサのための制御記憶は、内部および外部記憶の両方、すなわち即時アクセスのための32K内部SRAM (static random access memory: 静的ランダム・アクセス・メモリ) 28、高速アクセスのための外部ZBT (zero bus turnaround) SRAM 30、および大きい記憶要件のための外部DDR (double data rate) DRAM (dynamic random access memory: 動的ランダム・アクセス・メモリ) 32によって提供される。

【0022】接続された制御ポイント・プロセッサ34上で動作する組込みハードウェア・アクセラレータを、前処理アルゴリズムと共に用いると、ネットワーク・プロセッサ10は、複雑な範囲、優先順位、およびアクション仕様を有する百以上のフィルタ・ルールを介して、ワイヤスピードでフレームを処理することができる。これにより、ネットワーク・プロセッサに基づくシステムが、ゲートウェイ、サーバ・ファーム・アプリケーション、および混合トラヒックの処理に関連付けられたフィルタリング・タスクによく適合するようになる。

【0023】制御ポイント・ソフトウェアは、ネットワーク管理者がフィルタ・ルールを一貫性のあるユーザフレンドリなインタフェースにいつ入力するかをチェックする自動ロジックを提供する。安定性理論に基づく新規なフロー制御を用いて、ネットワーク・プロセッサ10は、TCP (Transmission Control Protocol: 伝送制御プロトコル) 崩壊することなく、通常使用されるランダム早期廃棄法 (random early discard method) より高い割合の一時的オーバーサブスクリプション (oversubscription) に耐える。ネットワーク・プロセッサ10は、帯域幅を自動的に割り振ることにより細分化されたサービスもまた配信し、瞬間的または仮定したトラヒック統計に基づいて数十のしきい値を設定することの効果予測しななければならないことからネットワーク管理者を解放する。

【0024】単一のネットワーク・プロセッサ10は、最大40個のFast Ethernetまたは4個のGigabit Ethernetポートに対するメディア・スピード・スイッチングを提供する。このネットワーク・プロセッサは、OC-48c、OC-48、4つのOC-12、または16個のOC-3ポートをサポートするように構成することもまたできる。スケーラビリティに関しては、2つのネットワーク・プロセッサを相互接続するために2つの3.5GbpsシリアルDASLリンクを使用して、その結果ポート密度を2倍にするか、またはスイッチ機構を接続して、最大64個のネットワーク・プロセッサを有するスイッチング・ソリューションを作り出すことができる。1つのプライマリおよび1つのセカンダリという、

2つのDASLリンクは、システム可用性を向上させるための冗長スイッチ機構に対する接続もまた提供することができる。

【0025】ネットワーク・プロセッサ10の一例示の実施形態は、図1に示される次の主要なセクションを含む。

1. 最大16個のプログラム可能プロセッサおよびコプロセッサを含む組込みプロセッサ複合体(EPC) 12。

2. Ethernet物理層デバイスからスイッチ機構に移動するフレームのためのエンキュー・デキュー・スケジューリング論理回路14(EDS流入)。

3. スwitch機構からEthernet物理層デバイスに移動するフレームのためのエンキュー・デキュー・スケジューリング論理回路16(EDS流出)。

4. 他のネットワーク・プロセッサまたは中間スイッチに対する相互接続のための流入スイッチ・インタフェース(スイッチ流入)18および流出スイッチ・インタフェース(スイッチ流出)20DASLリンク。

5. EthernetまたはPOS物理層デバイス26からフレームを受信する物理MACマルチプレクサ22(PMM流入)、およびEthernetまたはPOS物理層デバイス26にフレームを送信する物理MACマルチプレクサ24(PMM流出)。

【0026】図2は、組込みプロセッサ複合体の一例示の実施形態を示す。この組込みプロセッサ複合体は、2128MIPSの処理能力を提供する16個のプロトコル・プロセッサを含む。各プロトコル・プロセッサ40は、3ステージのパイプライン(フェッチ、デコードおよび実行)、汎用レジスタ、専用レジスタ、8命令キャッシュ、専用演算論理ユニット(ALU)およびすべて133MHzで動作するコプロセッサを含む。プロトコル・プロセッサの2つ、すなわち誘導フレームを処理するためのプロトコル・プロセッサ(guided frame handler: 誘導フレーム・ハンドラ)、および制御メモリ内のルックアップ・データを構築するためのプロトコル・プロセッサ(generic tree handler: 総称的ツリー・ハンドラ)は特別である。

【0027】図3は、プロトコル・プロセッサの一例示の実施形態を示す。プログラム可能プロトコル・プロセッサ40のそれぞれに関連付けられたコプロセッサは次の機能を提供する。

1. データ記憶コプロセッサ64は、フレーム・バッファ・メモリ42、44(流入/流出方向)のインタフェースとなり、直接メモリ・アクセス(DMA)機能を提供する。

2. チェックサム・コプロセッサ62は、ヘッダ・チェックサムを計算する。

3. エンキュー・コプロセッサ66は、256ビット・ワーク・レジスタに対するアクセスを制御し、キー・フ

レーム・パラメータを含む。このコプロセッサは、完了ユニット46とのインタフェースとなり、フレームをスイッチおよびターゲット・ポート・キューにエンキューする。

4. インタフェース・コプロセッサは、デバッグまたは統計収集のために、内部レジスタ、カウンタ、およびメモリに対するすべてのプロトコル・プロセッサ・アクセスを提供する。

5. 文字列コピー・コプロセッサは、EPC中のデータの効率的移動を可能にする。

6. カウンタ・コプロセッサは、プロトコル・プロセッサ40のためのカウンタ更新を管理する。

7. ポリシ・コプロセッサは、フロー制御情報を調査し、予め割り振られた帯域幅との一致をチェックする。

【0028】ハードウェア・アクセラレータ48は、フレーム転送、フレーム・フィルタリング、フレーム変更およびツリー検索を実行する。ネットワーク・プロセッサに組込まれた他の機能には、革新的なフィルタ・ルール処理、ハッシュ関数およびフロー制御が含まれる。

【0029】プロトコル・プロセッサ40は、複雑な範囲およびアクション仕様を有する百以上のフレーム・フィルタ・ルールを執行することができる。フィルタリングは、ネットワーク・セキュリティにとって本質的であり、ネットワーク・プロセッサ・ハードウェア支援48は、これらの複雑なルール・セットのワイヤスピードでの執行を提供する。フィルタ・ルールはフレームを拒否または許可するか、あるいはインターネット・プロトコル(IP)ヘッダ情報に基づいてサービスの品質(QoS)を割り振ることができる。前処理ルールのための制御ポイント・ソフトウェアは、ロジック・エラーを自動的に訂正する。論理的に正しいルール・セットが入力された後に、パケット・ヘッダ情報からキーが形成され、ネットワーク・プロセッサのソフトウェア管理ツリーを用いてワイヤスピードでテストされる。

【0030】理想的なランダム・ハッシュを上回るために、幾何学的ハッシュ関数はIPヘッダ内の統計的構造を利用する。その結果、衝突率の低下により、他の分解検索なしで、完全一致テーブル内の高速ルックアップが可能になる。

【0031】プロトコル・プロセッサの実行と並行して動作しながら、ツリー検索エンジン70は、(メモリ読取り、書込みまたは読み書きを含む)ツリー検索命令、メモリ範囲チェックおよび不正メモリ・アクセス通知を実行する。図14は、ツリー検索エンジンの一例示の実施形態を示す。

【0032】ネットワーク・プロセッサ10の中では2つのシステム制御オプションが利用可能である。内部プロセッサ34は、システムのための制御ポイント(CP)プロセッサとして機能することができ、あるいは外部プロセッサは、初期化および構成のための4つのEthe

metマクロの1つに接続することができる。CPプロセッサ34は、誘導フレームと呼ばれる特別なEthernetフレームを構築することによって、諸ネットワーク・プロセッサ中の他のプロセッサ・エンティティと通信する。誘導フレームは、DASLリンクを越えて他のデバイスに転送することができ、単一のEthernetポートに接続された1つのCPプロセッサが、そのサブシステム内に含まれたすべてのネットワーク・プロセッサ・デバイスと通信し制御することを可能にする。各ネットワーク・プロセッサの内部プロセッサ34もまた、別々の32ビットPCIバスを用いて通信することができる。

【0033】ネットワーク・プロセッサ10は普通、サブシステム・ボード上に常駐し、プロトコル層（すなわち、層2、層3、層4およびより高い層）フレーム処理を提供する。CPサブシステム内のCPプロセッサ34上で動作するソフトウェアは、管理およびルート発見機能を提供する。CPコード、プロトコル・プロセッサ上で動作するピココード、および誘導フレーム・ハンドラ上で動作するピココードは、このシステムの初期化、転送パスの維持、およびシステムの管理を提供する。分散システムとして、CPおよび各ネットワーク・プロセッサ・サブシステムは、効率および性能の向上のために誘導フレームを用いて並列動作し通信する複数のプロセッサを含む。

【0034】データ・フレームは、PMM22によってメディアから受信され、データ記憶バッファ42に転送される。PMMは、受信処理中にCRCチェックおよびフレーム妥当性検査もまた実行する。ディスパッチャ50は、最大64バイトのフレーム情報をフレーム・ルックアップに利用可能なプロトコル・プロセッサ40に送る。クラシファイヤ・ハードウェア支援48は、制御データを供給してフレーム・フォーマットを識別する。プロトコル・プロセッサ40は、固定一致ツリー、最長プレフィクス一致ツリー、ソフトウェア管理ツリーを含む、適用すべきツリー検索アルゴリズムを決定するために制御データを使用する。

【0035】ルックアップは、ツリー検索エンジン(TSE)70の支援で実行される。TSE70は、制御メモリ72のアクセスを実行し、プロトコル・プロセッサが実行を続けることを可能にする。制御メモリ72は、すべてのテーブル、カウンタおよびピココードに必要とされる他の任意のデータを記憶する。効率のために、制御メモリ・アービタ(arbiter)52は、プロトコル・プロセッサ40と様々なオンチップおよびオフチップ制御メモリ・オプション54の間にメモリ・サイクルを割り振ることによって制御メモリ動作を管理する。

【0036】プロトコル・プロセッサ40は、データ記憶動作のためのプライマリ・データ・バッファ、スクラッチ・パッド・データ・バッファおよび諸制御レジスタ(集合的に72)を含む。一致が見つかった、VLAN

ヘッダ挿入あるいはオーバーレイなどの流入フレーム変更を適用することができる。これらの変更は、EPC12によっては実行されない。代わりに、ハードウェア・フラグが設定されている場合、流入スイッチ・インタフェース・ハードウェア18が変更を実行する。他のフレーム変更は、流入データ記憶42の中に保持されるフレーム内容を修正することによってピココードおよびデータ記憶コプロセッサ64により達成することができる。

【0037】制御データは、スイッチ機構に対してフレームを送る前にスイッチ・ヘッダおよびフレーム・ヘッダを構築するために収集され使用される。制御データは、宛先ポートのフレーム・ルックアップ、マルチキャストまたはユニキャスト動作、および流出フレーム変更、の促進を助けるために、フレームの宛先などのスイッチ情報、ならびに流出ネットワーク・プロセッサのための情報を含む。

【0038】図4は、一例示的流入/流出フレーム・フローを示す。完了時にエンキュー・コプロセッサ66が、キュー制御ブロック(QCB)74にフレームをエンキューするために必要なフォーマットを構築し、そのフォーマットを完了ユニット46に転送する。完了ユニット46は、最大16個のプロトコル・プロセッサからスイッチ機構キュー76までのフレーム順序を保証する。スイッチ機構キューからのフレームは、スイッチ機構76により送信されるときに、スイッチ・ヘッダおよびフレーム・ヘッダ・バイトが挿入されて、64バイトのセルに分割される。

【0039】スイッチ機構76から受信されたフレームは、再組立て制御ブロック(RCB)80およびEDS流出44により提供された情報を用いて流出データ記憶バッファ78内に置かれ、EPC12にエンキューされる。フレームの一部は、フレーム・ルックアップを実行するために、ディスパッチャ50によって任意の遊んでいるプロトコル・プロセッサ40に送られる。フレーム・データは、クラシファイヤ・ハードウェア支援48からのデータとともにプロトコル・プロセッサ40に発送される。クラシファイヤ・ハードウェア支援48は、流入ネットワーク・プロセッサによって作り出されたフレーム制御データを用いて、流出処理のための開始命令アドレスを決定するのを助ける。

【0040】流出ツリー検索は、流入検索に対してサポートされたのと同じアルゴリズムをサポートする。TSE70でルックアップが実行され、実行を続けるためにプロトコル・プロセッサ40を解放する。すべての制御メモリ動作は、プロセッサ複合体間のメモリ・アクセスを割り振る制御メモリ・アービタ52によって管理される。

【0041】流出フレーム・データは、データ記憶コプロセッサ64を介してアクセスされる。正常なルックアップの結果は、転送情報、および場合によっては、フレ

ーム変更情報を含む。流出フレーム変更には、VLANヘッダ削除、存続時間 (time to live) 増分 (IPX) または減分 (IP)、IPヘッダ・チェックサム再計算、EthernetフレームCRCオーバーレイ、およびMAC宛先アドレスまたはソース・アドレスのオーバーレイまたは挿入を含むことができる。IPヘッダ・チェックサムは、チェックサム・コプロセッサ62によって準備される。変更は、組込みプロセッサ複合体12によって実行されるのではなく、むしろハードウェア・フラグが作り出されて、PMM流出ハードウェア24が変更を実行する。完了時に、EDS流出キュー44内のフレームをエンキューするために必要なフォーマットを構築するためにエンキュー・コプロセッサ46が用いられ、そのフォーマットを完了ユニット46に転送する。完了ユニット46は、最大16個のプロトコル・プロセッサから、流出Ethernet MACを供給するEDS流出キュー44のフレーム順序を保証する。完了したフレームは、最後にPMM流出ハードウェア24によって、Ethernet MACまたはPOSインタフェース、および物理ポートの外に送られる。

【0042】図14に示されたツリー検索エンジン (TSE) 70は、情報を記憶し取り出すためにツリー概念を使用する。取り出し、すなわちツリー検索ならびに挿入および削除は、例えばMACソース・アドレス、IPソース・アドレスとIP宛先アドレスの連結などのビット・パターンであるキーに基づいて行われる。本発明において使用する一例示的ツリー・データ構造100が図5に示されている。情報は、少なくともキー102を含むリーフ116、118、120、122と呼ばれる制御ブロック内に記憶される (記憶されるビット・パターンは、実際にはハッシュ化キー106である)。リーフは、エージング情報、ユーザ情報などの追加情報もまた含むことができ、このユーザ情報は、ターゲット・ブレード、ターゲット・ポート番号などの転送情報とすることができる。リーフのフォーマットは、ピココードによって定義され、内部または外部制御記憶にオブジェクトが置かれる。

【0043】ツリーに対する検索アルゴリズムは、キー102を含む入力パラメータに対して動作し、そのキーに対してハッシュ104を実行し、直接テーブル (DT) 108にアクセスし、パターン検索制御ブロック (PSCB) 110、112、114を介してツリーをウォークし、リーフ116、118、120、122で終了する。各タイプのツリーは、自分自身の検索アルゴリズムを有し、それにより異なるルールにしたがってツリーウォークが行われるようになる。例えば固定一致 (FM) ツリーに対しては、データ構造はパトリシア・ツリーである。リーフが見つかったとき、このリーフは入力キー102に一致することができる唯一の可能な候補である。「最終比較」動作が、入力キー102とリー

フ内に記憶されたパターンを比較する。これが、リーフが入力キー102に実際に一致するかどうかを検証する。この検索の結果は、リーフが見つかり一致したときには、成功 (OK) であり、他のすべての場合は失敗 (KO) である。

【0044】検索動作のための入力、次のパラメータを含む。

キー (key)

検索または挿入/削除の前に特別ピココード命令を用いて176ビット・キーが構築されなければならない。キー・レジスタは1つしかない。しかしツリー検索が開始された後には、TSE70が検索を実行しながら同時に、キー・レジスタを、次の検索のキーを構築するためにピココードによって使用することができる。これは、TSE70がキーをハッシュし、その結果を内部ハッシュ化キー・レジスタ106に記憶するからである。

キー長 (key length)

この8ビット・レジスタは、キー1ビットを含む。このレジスタは、キーの構築中にハードウェアによって自動的に更新される。

LUDefIndex

これは、検索が行われるツリーの完全な定義を含むルックアップ定義テーブル (LUDefTable) への8ビット・インデックスである。LUDefTableの内部構造は、図11に示されている。

TSRNr

検索結果は、1ビットのツリー検索結果領域TSROまたはTSR1のいずれかに記憶することができる。これはTSRNrによって指定される。TSEが検索している間に、ピココードは他のTSRにアクセスして前の検索結果を分析することができる。

カラー (color)

カラーがイネーブルになったツリー (LUDefTable内で指定される) に対して、16ビットのカラー・レジスタ124の内容が、ハッシュ動作中にキーの中に挿入される。

【0045】ルックアップ定義テーブルは、ツリー検索メモリを管理するメイン構造である。LUDefTableは、内部メモリ構造であり、ツリーを作り出すための128エントリを含む。LUDefTableは、ツリーが中に存在する物理メモリ (例えば、DRAM、SRAM、内部RAM)、キャッシュがイネーブルであるかどうか、キーおよびリーフのサイズ、ならびに実行すべき検索アクションのタイプ、を定義するエントリを含む。LUDefTableは、3つの別々のランダム・アクセス・メモリ、すなわち汎用プロセッサ・ツリー・ハンドラ (GTH) によってしかアクセス可能でない1つのRAM、および互いの複製であり、すべてのピコプロセッサによってアクセス可能な2つのRAMとして実施される。

【0046】ハッシュ関数104の出力はいつも、元の

入力キー102とハッシュ関数104の出力の間に1対1対応があるという特性を有する176ビットの数である。以下に説明するように、この特性は直接テーブル108の後に開始するツリーの深さを最小化する。

【0047】ツリーのためのカラーがイネーブルである、図5の例の場合、16ビットのカラー・レジスタ124が176ビットのハッシュ関数出力に挿入され、フィールドの結果は、HashedKey106と呼ばれる192ビットの数である。挿入は、直接テーブル108のすぐ後ろに行われる。直接テーブル108が2^Nのエントリを含む場合、図5に示すように、ビット位置Nに16ビットのカラー値が挿入される。ハッシュ関数の出力は、挿入されたカラー値と共に、HashedKeyレジスタ106内に記憶される。ツリーに対するカラーがディセーブルである場合、176ビットのハッシュ関数は修正しないで取られ、16個の0がハッシュ出力に追加されて、192ビットの最終HashedKeyを生成する。

【0048】カラーは、複数の独立したツリー間で単一の直接テーブル108を共用するために使用することができる。例えばカラーの使用例は、MACソース・アドレス(SA)テーブル内の仮想ローカル・エリア・ネットワーク(VLAN)である。この場合、入力キー102はMAC SAであり、カラー124はVLAN IDである(VLAN IDは12ビットなので、カラーの4ビットは使用されない、すなわち0に設定される)。ハッシュ関数104の後に使用されているパターンは、48+16=64ビットである。カラーは、もはやパターンの一部であり、異なるVLANのMACアドレスを区別する。

【0049】ハッシュ関数104は、自分の出力内のほとんどのエントロピが最高位の諸ビットにあるように定義される。HashedKeyレジスタ106の最高位からのNビットが、直接テーブル(DT)108へのインデックスを計算するために使用される。

【0050】SMTツリーに対して、入力キーおよびカラーは合わせて192ビットの入力パターンを形成する。この観点では、カラー・レジスタはキー・レジスタに対する拡張と見るべきである。SMTツリーは、

「0」に設定したカラー・イネーブル・ビットを有しなければならない。さらにSMTツリーは、16ビットのカラー・レジスタ124と共に176ビットの入力キー102を取り、192ビットのHashedKey106を生成するハッシュ関数を使用しなければならない、それによって、キーが最も左の176ビットを形成し、カラーが最も右の16ビットを形成する。したがってSMTツリーのためのハッシュ関数は、実際にはハッシュ関数ではなく、カラーをキーの拡張として使用する手段である。

【0051】記憶および検索効率を達成するために、この実施例は次のデータ構造を利用する。

a. 検索する必要のあるパターン/キー

- b. 直接テーブル(DT)エントリ
- c. パターン検索制御ブロック(PSCB)
- d. リーフ
- e. 最終比較動作
- f. 比較テーブル・エントリ

【0052】DTエントリは、キーの最初の「N」ビットに基づく最初のアドレス・ロケーションである。このエントリは、以下に述べる3つの部分を含む。PSCBエントリは中間ノード位置である。リーフ・エントリは、検索結果のためのアドレス・ロケーションである。比較テーブル・エントリは、リーフ/パターン比較パラメータを記述する。DTエントリは、以下にさらに述べるように、幅1かつ高さ1、または幅1かつ高さ2のいずれかにより定義される形を有する。

【0053】PSCBは、ツリー内の分岐である。この好ましい実施形態には、0分岐および1分岐がある。PSCBから出る分岐の数は、分岐を指定するために使用されるビットの数に依存する変数である。nビットが使用される場合には、PSCBで2ⁿの分岐が定義される。各PSCBは、ビット位置pにもまたに関連付けられる。PSCBから0分岐を介して到達することができるすべてのリーフは、パターン中の位置pに「0」を有し、1分岐を介して到達することができるリーフは、位置pに「1」を有する。さらにPSCBから到達することができるすべてのリーフは、ビット0...p-1が同一のパターンを常に有する、すなわちこのパターンは、位置pにおいて始めて変化する。PSCBに関連付けられたビット位置は、前のPSCBまたはDTエントリ内に記憶され、NBT(すなわちnext bit to test: テストすべき次のビット)と呼ばれる。

【0054】したがってPSCBは、ツリー内のリーフ・パターンが変化する位置にしか挿入されない。PSCBの数、したがって検索性能は、ツリー内のリーフ数にのみ依存し、パターンの長さには依存しないので、これにより効率的検索動作が可能になる。PSCBレジスタ・フォーマットは、図12に示されている。

【0055】SMTツリーにおいてPSCBの後の第1リーフは、LUDefTable内に定義された形を有しなければならない。リーフ連鎖内の他の任意のリーフは、連鎖ポインタ内の5ビット、すなわちリーフ内のNLASMTフィールド、により定義される形を有する。

【0056】DTおよびPSCBエントリのフォーマットは、同一であり、次の部分を含む。

- a. SCB(search control block: 検索制御ブロック) 2ビット
- b. NPA(next pattern address: 次のパターン・アドレス): 次のPSCBアドレスを指す、またはLCBA(leaf control block address: リーフ制御ブロック・アドレス): リーフ/結果を指す。
- c. NBT(next bit or bits to test: テストすべき

21

次のビット) : テストすべき次の1対または1組の「x」ビット (x=1またはn) とすることができる。テストすべきビット数は、記憶効率に基づき決定される。

d. 直接リーフ。

【0057】この実施例の各エントリは、幅36ビットであり、現在定義されている3つの可能なエントリ・フォーマットの1つを含む。

a. 空のDTエントリ: SCB=00かつNPA=0かつNBTが不正または0。

b. NPA/NBTが妥当: SCB=00かつNPAおよびNBTが妥当。DTエントリに対して、NPAは第1中間ノードを指し、NBTはテストすべき1ビットまたは複数ビットを指す。PSCBエントリの場合、NPAは、このトレイル内の他のノードを指す。

c. LCBAが妥当: SCB=01。LCBAは、関連付けられたリーフ・アドレス、すなわち検索結果を指す。

d. 直接リーフ: SCB=10かつデータの残りは検索結果またはリーフを含む。リーフ・データの一部は、大きな検索結果記憶をサポートするために、リーフ・アドレスの連鎖を含むことができる。

【0058】DTかつPSCBエントリのメモリ割振りに関して: 常に2**テストすべきビット数個のアドレスからなる、すなわち対/組であるということを除き、SMT PSCBはSMT DTエントリと同じ構造を有する。これらのアドレス対または組は、メモリ内に連続的に割り振られ、検索ツリーをウォークするための分岐/ジャンプ・ポインタとして使用される。

【0059】SMTツリー内のリーフのフォーマットは、2つのパターンを含む制御情報を含む。この2つのパターンは、範囲比較を定義するために使用される。SMT内のリーフは、NLASMTフィールドを用いて結びつけることができる。図13は、SMTツリーのためのフォーマットを示す。PSCBの後の第1リーフに到達するとき、最終比較動作が実行される。これがOKを戻すとき、検索は停止する。しかし最終比較がKOを戻し、非0のNLASMTフィールドがあるとき、次のリーフが読み込まれ、もう1度最終比較動作が実行される。この処理は、最終比較がOKを戻すか、またはNLASMTフィールドが0に等しくなるまで続き、この場合検索はKOで戻る。

【0060】ソフトウェア管理ツリー検索のための高レベル・アルゴリズム・フローは、次の通りである。

1. DTエントリ読み込み

a. SCB=00かつNPAおよびNBTが妥当である場合、NPAおよびNBTを読み込み、次のPSCBアドレスを生成する。

b. NBTが妥当でなく、直接リーフが妥当である場合、リーフ内容を読み込みリーフ評価ステップに行く。

22

c. NBTが妥当でなく、かつ/またはリーフが存在しない場合、KO、すなわち検索結果が失敗かつ完了フラグが実行済み、を戻す。

2. リーフ評価: パターン (キー) とリーフ内に記憶されたパターンを比較する。SMTは、常に2つのパターンを含む。これらの2つのパターンは、範囲比較を定義するために使用される。また、SMT内のリーフは、結びつけることもできる (すなわち次のリーフ・アドレスを表すNLASMTを用いる)。PSCBの後の第1リーフがヒットすると、最終比較動作が実行される。これがOK (成功) を戻すとき、検索はOKで停止する。しかしこの最終比較がKO (失敗) を戻し、非0の、すなわち妥当なNLASMTフィールドがあるとき、次のリーフが読み込まれ最終比較動作がもう1度実行される。この動作は、最終比較がOK (成功) で戻るか、またはNLASMTフィールドがもう妥当でなくなるまで続き、この場合、検索はKO (失敗) で戻る。

【0061】本明細書で述べるビット/レジスタ幅の値は、例示的であり、利用可能なメモリ、性能要件等を最適化するために異なる値に変更することができる。

【0062】検索は、直接テーブル108に対するアクセスで開始する、すなわち直接テーブルからDTエントリが読み込まれる。DTエントリを読み込むために使用されるアドレスは、レジスタ106内のHashedKeyの上位Nビットから、LUDefTable内で定義されたツリー特性に対して計算される。DTエントリは、ツリーのルートと見ることができる。実際のツリー・データ構造は、ツリー・タイプに依存する。SMTツリーに対するパトリシア・ツリー・データ構造への拡張が使用される。

【0063】8エントリDT108を使用する例が図6に示されている。検索時間すなわちアクセスされなければならないPSCBの数は、DTを用いることによって少なくすることができる。したがってDTサイズを大きくすることにより、メモリ使用量と検索性能の間のトレードオフが起こり得る。

【0064】DTエントリを読み込んだ結果、そのエントリが1リーフに対するポインタを含むだけで、その後そのリーフ自体を読み込まなければならないことになると、性能の理由で非効率である。この状況は、DTエントリにつき単一のリーフ・エントリを多数有するFMツリーに対して、非常にしばしば発生する。直接リーフのコンセプトは、より多くのメモリ使用とよりよい性能とのトレードオフを可能にする。

【0065】ツリーは、直接リーフをイネーブルにすることができ、これはルックアップ定義テーブル (LUDefTable) において指定される。直接リーフをイネーブルにしたツリーとディセーブルにしたツリーの間の相違は、図7に示されている。直接リーフがイネーブルで、DTエントリが単一のリーフを含むとき、このリーフは、DTエントリ自体に直接記憶される。そうでない場合、こ

のDTエントリは、リーフに対するポインタを含む。

【0066】整形は、TSMの一機能であり、リーフ、PSCBのようなオブジェクトがTSM内にどのように記憶されるかを指定するために用いられる。形は、幅および高さのパラメータによって定義される。オブジェクトの高さは、オブジェクトが記憶される連続的アドレス・ロケーションの数を示す。オブジェクトの幅は、オブジェクトが記憶される連続的バンクの数を示す。幅および高さに対して、ハードウェアは適切な数のロケーションを自動的に読み込む。ピココードの視点からは、オブジェクトはアクセスの原子単位である。SRAM内に記憶されるオブジェクトに対する幅は、常に1でなければならない。DRAM内のオブジェクトに対する幅は、1より大きくてよい。単一メモリ・ロケーション中に適合するほど充分小さいオブジェクトは、高さ1、幅1を有すると定義される。直接リーフがディセーブルにされたDTエントリの形は、常に(W=1、H=1)である。DTエントリがDRAM(動的ランダム・アクセス・メモリ)内に記憶されるとき、このエントリは正確に64ビットを占める。直接リーフがイネーブルにされたDTエントリの形は、LUDefTable内に指定されたリーフの形に等しい。一般にこれより、DT108によってより多くのメモリが使用されることになる。これはまた、リーフの形がDTエントリ・アドレス計算に対して影響を与えることにもなる。

【0067】DTエントリが読み込まれた後、このDTエントリが直接リーフを含まず、また空でもないかと仮定すると、このDTエントリで始まるツリーをウォークすることにより検索が続く。ツリーウォークは、リーフに到達するまで、いくつかのPSCB(パターン検索制御ブロック)を通過することができる。

【0068】SMTツリー内の検索中にPSCBに出会うと、ツリー検索エンジン・ハードウェアは、HashedKeyのビットpの値次第で、0分岐または1分岐に対してツリーウォークを続ける。

【0069】図10は、本発明のソフトウェア管理ツリー検索アルゴリズムの処理ロジックを示す。アルゴリズムは、ロジック・ブロック1100において入力キーの読み込みで始まる。SMT検索では、入力キーをハッシュ化キーにハッシュするオプションがある。ハッシュ関数は、入力キー内のビット順序を反転することができる。例えばビット1とビット7を交換することができる。ハッシュ関数が使用されるとき、このハッシュ関数は、プログラム可能、すなわちどのビットが交換されるかについてプログラムすることができる。ロジック・ブロック1002により示されるように、直接テーブルが次に読み込まれる。ハッシュ化キーの上位Nビット(ここでNは構成可能である)が直接テーブルへのインデックスとして使用される。読み込まれたエントリが空であるとき、終了ブロック1004により示されるように、検索はKO

(発見なし)を戻す。このエントリが判定ブロック1006内のリーフを指す場合、処理は、ロジック・ブロック1010においてリーフの内容を読み込むことで続く。そうでない場合、エントリはPSCBを指し、ロジック・ブロック1008に示されるように、PSCBの適切な部分が読み込まれる。SMT検索に関して、PSCBはFM PSCBのフォーマットを有することができる。あるいは以下に述べるように、より高度なフォーマットを有することができる。

【0070】ロジック・ブロック1008から、処理は判定ブロック1006に戻り、リーフが見つかったかどうかを判定する。判定ブロック1006においてリーフが見つかるとき、ロジック1010によって示されるように、このリーフが読み込まれ、次いでロジック・ブロック1012によって示されるように、入力キーと比較される。判定ブロック1014においてリーフ内に記憶されたパターンと入力キーの間の一致がある場合、検索はOK(成功)を戻し、終了ブロック1018によって示されるように、リーフの内容をアプリケーションに渡す。判定ブロック1014において一致がない場合、判定ブロック1016において示されるように、次のリーフが存在するかどうかの判定で処理が続く。リーフ連鎖内に次のリーフがある場合、処理はロジック・ブロック1010に戻ってそのリーフの内容を読み込む。そうではなくて、次のリーフがない場合、終了ブロック1020によって示されるように、検索はKO(失敗)を戻す。

【0071】リーフ・パターンとハッシュ化キーの間の比較動作は、FMとLPM検索よりも複雑である。リーフは、2つのパターンp1およびp2を含む。p1およびp2内のハッシュ化キーは、それぞれN個のフィールドに論理的に分割される。この比較動作は、N個の部分比較を含み、全体の比較がOKを戻すためには、すべてがOK(成功)を戻さなければならない。それぞれの部分比較は、(i)p1が最小、p2が最大を表す(i)p1がパターンを表し、p2がマスクを表すという2つのモードにしたがって実行することができる。第1の場合、部分比較は $p1 \leq \text{ハッシュ化キー} \leq p2$ のときOKを戻す。第2の場合、部分比較は、 $(\text{ハッシュ化キー} \text{ AND } p2) = (p1 \text{ AND } p2)$ のときOKを戻す。

【0072】フィールドの定義(各フィールドのサイズおよび位置)および各フィールドに対する比較モードは、リーフ内にコード化された形態で記憶するか、特別なルックアップ・テーブル内に記憶することができる。この好ましい実施形態では、CompDefTableが使用され、ここでテーブルのインデックスはリーフ内に記憶される。

【0073】拡張として、PSCBは2^b個のエントリから構成することができ、その結果PSCBからどのエントリを読み込むかをハッシュ化キーのbビットが選択す

る。これは、より多くのメモリ使用を犠牲にして性能を向上させる。さらにPSCBは、リーフ比較で述べたのと同じ方法で動作する1つまたは2つのパターン(p1およびp2)を各エントリが含むように拡張することができる。例えばPSCBが、これもまたPSCB内に記憶された長さLのp1およびp2を含むと仮定する。その場合、ハッシュ化キーから、NBTによって与えられる位置のLビットが取られる。これらのLビットは整数Iとして解釈される。 $p1 \leq I \leq p2$ のとき、次のPSCBのエントリIが使用され、そうでない場合、次のPSCBのエントリ0が使用される。

【0074】ツリー内の検索性能をよくするためにキャッシュを使用することができる。キャッシュの使用は、LUDefTable内でツリー毎にイネーブルにすることができる。検索中に、ツリー検索エンジン70は、キャッシュ内を最初にチェックし、HashedKeyと一致するリーフが存在するかどうかを判定する。そのようなリーフが見つかる場合、そのリーフが戻され、これ以上の検索は必要でない。そのようなリーフが見つからない場合、通常の実行が始まる。

【0075】ツリー検索エンジン・ハードウェア70に対して、キャッシュ・ルックアップは、通常の実行と正確に同一である。したがって入力キーがHashedKeyにハッシュ化され、直接テーブル108アクセスが実行される。直接テーブル108はキャッシュとしてふるまう。キャッシュ検索がOK(成功)を戻すとき、検索は終了する。そうでない場合、ツリー検索エンジンは、ハッシュ動作が実行されないということを除き、ツリー全体の第2検索を開始する。HashedKeyレジスタ106の内容は、再利用される。

【0076】キャッシュ検索を使用するかどうかは、LUDefTable内で指定することができる。キャッシュ検索がLUDefTableエントリIを使用し、検索がKO(失敗)で終了する場合、LUDefTableエントリI+1を用いた追加の実行が自動的に始まる。原則的には、これにより複数の検索を結びつけることが可能になるが、LUDefTableエントリI+1の下でツリー全体を記憶することが推奨される。

【0077】ツリー検索エンジン70は、FMツリー、LPMツリーおよびSMTツリーにおいてハードウェア40 検索動作を提供する。すべてのツリー・タイプに対して、ツリーを初期化し維持するために様々な量のソフトウェアが必要とされる。FMツリーおよびLPMツリーだけが、制御ポイント・プロセッサ34の介入なしに、リーフの挿入および削除を行うことができる。この機能を用いることにより、スケラブルな構成が可能になり、さらに、必要であれば、CP34がリーフを挿入または削除することが可能になる柔軟性ができる。

【0078】SMTは、CPによって定義された検索機構に従うツリー構造をCPが作り出す機構を提供する。50

この一例は、IPソース・アドレス(IPSA)、IP宛先アドレス(IPDA)、ソース・ポート、宛先ポートおよびプロトコルを含むIP5タプル・フィルタリング・テーブルである。FMおよびLPMツリーとは対照的に、SMTは、範囲に対するサポートを提供する。例えば、ソース・ポートが範囲100...110以内になければならないことを指定するために1リーフを用いることができる。

【0079】SMTとFM/LPMツリーの間には次の相違が存在する。

- ・SMTは、リーフ内に2つのパターンを常に含む。この2つのパターンは、以下に説明する範囲比較を定義するために使用される。

- ・SMT内のリーフは、NLASMT(次のリーフ・アドレス)フィールドを用いて結びつけることができる。PSCB後の第1リーフがヒットするとき、最終比較動作が実行される。これがOK(成功)を戻すとき、検索はOKで停止する。しかしこれがKO(失敗)を戻し、非0のNLASMTフィールドがあるとき、次のリーフが読み込まれ、最終比較動作がもう1度実行される。この処理は、最終比較がOK(成功)を戻すか、またはNLASMTフィールドが0に等しくなるかのいずれかまで続き、この場合、検索は、KO(失敗)を戻す。

【0080】FMおよびLPMツリーとは対照的に、SMTは各リーフ内に同じ長さの2つのパターンを含む。比較動作の目的で、入力キー(および同様に、リーフ内に記憶された2つのパターン)は、複数のフィールドに論理的に分割することができる。一例が図8に示されている。各フィールドに対して、2つの比較のうち1つを実行することができる。

1. マスク下の比較。入力キー中のビットが、リーフ・パターン1に指定されたマスクの下で、リーフ・パターン0の中のビットと比較される。マスク中の「1」は、入力キー中の対応するビットが、パターン0の中の対応するビットに等しくなければならないことを表し、マスク中の「0」は、入力キー中の対応するビットが、比較に何も影響を与えないことを表す。すべてのビット比較がOKの場合に限り、フィールド全体の比較がOKになる。

2. 範囲下の比較。入力キー中のビットは、整数として扱われ、この整数がチェックされて、最小と最大(両者とも範囲に含まれる)によって与えられる範囲内にあるかどうか判定される。この場合、フィールド比較はOKであり、そうでない場合、フィールド比較はKOである。すべてのフィールド比較がOKの場合に限り、最終比較全体がOKを戻し、そうでない場合、最終比較は、KOを戻す。

【0081】論理フィールドがどのように定義されるかの定義は、比較定義テーブル(CompDefTable)内に指定され、そのエントリ・フォーマットの一例が図9に与え

られている。省略された場合、CompDefTable内に反対を指定するエントリがない限り、フィールドはマスク下の比較フィールドである。

【0082】CompDefTable内の各エントリは、1つまたは2つの範囲比較を指定する。以下に説明するように、2つより多い範囲比較を指定するために、複数のエントリを使用することができる。各範囲比較は、2つのパラメータによって定義される。

・フィールドの第1ビットの位置であるオフセット。オフセットは、16ビット境界になければならず、次の0、16、32、48、64、80、96、112、128の値を取ることができる。

・ビット内のフィールド長。長さは、次の8、16、24、32の値を取ることができる。

【0083】例えば、図8に示されたキーに対して、ソース・ポート・フィールドに対する範囲下の比較であれば、64に設定されたOffset0および16に設定されたMin/MaxLength0を有し、宛先ポート・フィールドに対する範囲下の比較であれば、80に設定されたOffset1および16に設定されたMin/MaxLength1を有する。2つより多い範囲比較が必要である場合、継続ビットが1に設定されなければならず、これにより、CompDefTable内の次のエントリは、追加の1つまたは2つの範囲定義下の比較のために使用されるようになる。SMT比較のために使用されるCompDefTable内のインデックスは、リーフ内に指定される。

【0084】性能の理由から、範囲下の比較を可能な限り使用しないことが推奨される。範囲下の比較はそれぞれ、実行するのに1クロック・サイクル（7.5ナノ秒）余分にかかる。したがって範囲が2の累乗（すなわち128～255）の場合、範囲下の比較は必要でなく、この種の範囲はマスク下の比較動作を用いて処理することができる。

【0085】SMT最終比較が失敗し、リーフ内のNLASMTフィールドが非0であるとき、TSE70は、比較がOKを戻すか、NLASMTが0に等しくなるまで、次のリーフを読み込み、最終比較動作をもう1度実行する。

【0086】本発明は、ハードウェア、ソフトウェア、または両者の組合わせで実現することができる。本明細書に述べた方法を実行するように構成された任意の種類のコンピュータ・システムまたは他の装置が適合する。ハードウェアとソフトウェアの一典型的組合わせは、ロードされ実行されたときに、本明細書に述べた方法を実行するようにコンピュータ・システムを制御する汎用コンピュータ・システムである。本発明は、本明細書に述べた方法の実施を可能にするすべての機能を備え、コンピュータ・システムにロードされるときにこれらの方法を実行することができるコンピュータ・プログラム・プロダクトに組み込むこともまたできる。

【0087】本文脈内のコンピュータ・プログラム命令またはコンピュータ・プログラムは、情報処理機能を有するシステムに、直接、または次のa)他の言語、コードまたは表記への変換、b)異なる材質形態での再生産、のいずれか一方または両方が行われた後に特定の機能を実行させることを意図する1組の命令の、任意の言語、コード（すなわちビココード命令）または表記での任意の表現を意味する。

【0088】本発明の精神および範囲を逸脱することなく、本発明の好ましい実施形態に対して多くの修正が可能であることを、当業者は理解するであろう。さらに本発明のいくつかの機能を、他の機能を対応して使用することなく使用することが可能である。したがって、本発明の範囲は付随の請求の範囲によってのみ定義されるので、好ましい実施形態の前記記述は、本発明の原理を示す目的で提供され、その制限内にはない。

【0089】まとめとして、本発明の構成に関して以下の事項を開示する。

【0090】(1) ソフトウェア管理ツリー内の可変長検索キーに対して、コンピュータ処理デバイスによりパターン範囲比較を実行する方法であって、検索文字列として入力キーを読み込む行為と、検索キーの最上位Nビットを、検索ツリーの複数のルート・ノードを表すテーブルに対するインデックスとして使用する行為であって、非空のエントリのそれぞれが検索ツリー内の次の分岐またはリーフに対するポインタを含む行為と、非空のテーブル・エントリ内の前記ポインタが、対応する検索ツリーのリーフまたは次の分岐のどちらを指すかを判定する行為と、前記ポインタが前記対応する検索ツリーのリーフを指さない場合に、次の分岐内容を読み込む行為と、対応する検索ツリーのリーフに到達したときにリーフ内容を読み込み、前記リーフ内の一対のパターンと前記検索キーを比較して、前記一対のリーフ・パターンによって定義された範囲が前記検索キーを含むかどうかを判定する行為と、前記リーフ・パターンが前記検索キーを含む場合に、見つかったリーフの内容を、要求するアプリケーションに戻す行為とを含む方法。

(2) 検索キーを形成するためにプログラム可能ハッシュ関数を使用して入力キーをハッシュすることをさらに含む、上記(1)に記載のパターン範囲比較を実行する方法。

(3) 検索ツリーの複数のルート・ノードを表す前記テーブルが2個のエントリを含む、上記(1)に記載のパターン範囲比較を実行する方法。

(4) 前記コンピュータ処理デバイスがネットワーク・プロセッサである、上記(1)に記載のパターン範囲比較を実行する方法。

(5) 前記対応する検索ツリーの前記次の分岐の内容が他の次の分岐を指す、上記(1)に記載のパターン範囲比較を実行する方法。

(6) 前記次の分岐の内容が前記対応する検索ツリーの
前記リーフを指す、上記(1)に記載のパターン範囲比
較を実行する方法。

(7) 前記リーフ・パターンが前記検索キーを含まず、
他のリーフに対するポインタを含まない場合に、成功表
示を戻さないことをさらに含む、上記(1)に記載のパ
ターン範囲比較を実行する方法。

(8) 前記テーブルに対する前記インデックスが空エン
トリを指している場合に、成功表示を戻さないことをさ
らに含む、上記(1)に記載のパターン範囲比較を実行
する方法。 10

(9) 前記検索キーにカラー・レジスタの内容を追加し
て最終検索キーを提供することをさらに含む、上記

(1)に記載のパターン範囲比較を実行する方法。

(10) 前記検索キーに0の文字列を追加して最終検索
キーを提供することをさらに含む、上記(1)に記載の
パターン範囲比較を実行する方法。

(11) 前記一对のパターンによって定義される範囲内
にあるかどうかを判定するためにチェックされる整数と
して前記検索キー内の諸ビットが扱われる、範囲下の比
較動作を、一对のパターンを比較する前記行為が含む、 20
上記(1)に記載のパターン範囲比較を実行する方法。

(12) 第2リーフ・パターン内に指定されたマスク下
で第1リーフ・パターン内の諸ビットと前記検索キー内
の諸ビットが比較される、マスク下の比較動作を、一对
のパターンを比較する前記行為が含む、上記(1)に記
載の方法。

(13) 前記リーフが他のリーフに対する連鎖ポインタ
を含む場合に、他のリーフ内に記憶された一对のパター
ンを読み込み、前記パターンと前記検索キーを比較する行
為と、記憶された前記パターンが前記検索キーを含ま
ず、連鎖内の次のリーフに対するポインタを含まない場
合に、成功表示を戻さない行為とをさらに含む、上記 30

(1)に記載のパターン範囲比較を実行する方法。

(14) 前記リーフが他のリーフに対する連鎖ポインタ
を含む場合に、他のリーフに記憶された一对のパターン
を読み込み、前記パターンと前記検索キーを比較する行為
と、記憶された前記パターンが前記検索キーを含む場合
に、成功表示を戻す行為とをさらに含む、上記(1)に
記載のパターン範囲比較を実行する方法。 40

(15) ソフトウェア管理ツリー内の可変長検索キーに
対してパターン範囲比較を実行するための複数のデータ
構造を含むコンピュータ可読媒体であって、検索される
べきパターンまたはキーと、検索ツリーに対する第1ア
ドレス・ロケーションを記憶する直接テーブルと、前記
検索ツリー内の分岐をそれぞれ表す複数のパターン検索
制御ブロックと、各エントリに関連付けられた少なくと
も1つの範囲比較を指定する比較テーブルと、複数のリー
フであって、前記検索キーと比較する一对のパターン
を各リーフが記憶するリーフとを含むコンピュータ可読 50

媒体。

(16) ツリー検索メモリを管理するルックアップ定義
テーブルをさらに含む、上記(15)に記載の、パター
ン範囲比較を実行するための複数のデータ構造を含むコ
ンピュータ可読媒体。

(17) 前記ルックアップ定義テーブルが、前記ツリー
が中にある物理メモリ、前記キーおよびリーフのサイ
ズ、ならびに実行されるべき検索のタイプ、を定義する
エントリを含む、上記(15)に記載の、パターン範囲
比較を実行するための複数のデータ構造を含むコンピュ
ータ可読媒体。

(18) 前記ルックアップ定義テーブルが、複数のメモ
リに実施される、上記(15)に記載の、パターン範囲
比較を実行するための複数のデータ構造を含むコンピュ
ータ可読媒体。

(19) 直接テーブル・エントリのフォーマットが、少
なくとも1つの検索制御ブロック、次のパターン検索制
御ブロックを指す次のパターン・アドレス、リーフまた
は結果を指すリーフ制御ブロック・アドレス、テストす
べき次のビットまたは諸ビット、および直接リーフを含
む、上記(15)に記載の、パターン範囲比較を実行す
るための複数のデータ構造を含むコンピュータ可読媒
体。

(20) パターン検索制御ブロックのフォーマットが、
少なくとも1つの検索制御ブロック、次のパターン検索
制御ブロックを指す次のパターン・アドレス、リーフま
たは結果を指すリーフ制御ブロック・アドレス、および
テストすべき次のビットまたは諸ビットを含む、上記

(15)に記載の、パターン範囲比較を実行するための
複数のデータ構造を含むコンピュータ可読媒体。

(21) 前記比較テーブルが、少なくとも1つの範囲比
較を定義するエントリを含み、各範囲比較が、前記フィ
ールドの第1ビットの位置であるオフセット・パラメー
タおよび前記フィールドのビット長である長さパラメー
タによって定義される、上記(15)に記載の、パター
ン範囲比較を実行するための複数のデータ構造を含むコ
ンピュータ可読媒体。

(22) リーフ・データ構造が、少なくとも1つのリー
フ連鎖ポインタ、プレフィクス長、前記検索キーと比較
すべき一对のパターン、および可変ユーザ・データを含
む、上記(15)に記載の、パターン範囲比較を実行す
るための複数のデータ構造を含むコンピュータ可読媒
体。

(23) 前記直接リーフが、直接テーブル・エントリに
直接記憶され、検索制御ブロック、および検索キーと比
較すべき一对のパターンを含む、上記(15)に記載
の、パターン範囲比較を実行するための複数のデータ構
造を含むコンピュータ可読媒体。

(24) パターン検索制御ブロックが、前記検索ツリー
内のリーフ・パターンが変化する位置に挿入される、上

記(15)に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

(25) パターン検索制御ブロックが、幅1および高さ1により定義される形を有し、少なくとも36ビットのライン長を有するメモリ内に記憶される、上記(15)に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

(26) ソフトウェア管理ツリー内の可変長検索キーに対してパターン範囲比較を実行するための、半導体基板上に組立てられた装置であって、複数のプロトコル・プロセッサ、およびフレーム処理を提供する内部制御ポイント・プロセッサを含む組み込みプロセッサ複合体と、各プロトコル・プロセッサにアクセス可能で、高速パターン検索、データ操作、およびフレーム分析を提供する複数のハードウェア・アクセラレータ・コプロセッサと、少なくとも1つの検索ツリーを表す複数のデータ構造を記憶する複数のプログラム可能メモリ・デバイスであって、前記データ構造が、直接テーブル、パターン検索制御ブロック、比較テーブル、および前記検索キーと比較すべき一对のパターンを含むリーフを含むプログラム可能メモリ・デバイスと、前記複数のメモリ・デバイスに対する各プロトコル・プロセッサのアクセスを制御する制御メモリ・アービタとを備える装置。

(27) メモリ読み込みおよび書き込みならびにメモリ範囲チェックを含むツリー検索命令を実行するプロトコル・プロセッサの実行と並行して動作するツリー検索エンジンをさらに備える、上記(26)に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

(28) 前記複数のメモリ・デバイスが、少なくとも1つの内部静的ランダム・アクセス・メモリ、外部静的ランダム・アクセス・メモリ、および外部動的ランダム・アクセス・メモリをさらに備える、上記(26)に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

(29) 前記制御メモリ・アービタが、前記複数のプロトコル・プロセッサと前記複数のメモリ・デバイスの間でメモリ・サイクルを割り振ることによって制御メモリ動作を管理する、上記(26)に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

(30) 各プロトコル・プロセッサが、プライマリ・データ・バッファ、スクラッチ・パッド・データ・バッファおよびデータ記憶動作のための制御レジスタを備える、上記(26)に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

(31) 前記検索キーに対して幾何学的ハッシュ関数を実行するハッシュ・ボックス・コンポーネントをさらに備える、上記(26)に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

(32) プログラム可能検索キー・レジスタおよびプログラム可能ハッシュ化キー・レジスタをさらに備える、上記(26)に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

(33) 複数の独立した検索ツリー間で単一のテーブル・データ構造を共用することを可能にするためのプログラム可能カラー・キー・レジスタをさらに備える、上記(32)に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

(34) 前記カラー・レジスタの内容が、もしイネーブルであれば、前記ハッシュ出力に追加されて最終ハッシュ化キーを生成する、上記(33)に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

(35) 前記カラー・レジスタがイネーブルでない場合に、最終ハッシュ化キー生成のために前記ハッシュ出力に同数の0を追加する、上記(33)に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

(36) ソフトウェア管理ツリー内の可変長検索キーに対してパターン範囲比較を実行するコンピュータ・プログラム・プロダクトを含むコンピュータ可読媒体であって、入力キーを検索文字列として読み込むプログラム命令と、前記検索キーの最上位Nビットを、検索ツリーの複数のルート・ノードを表すテーブルに対するインデックスとして使用するプログラム命令であって、それぞれの非空のエントリが、前記検索ツリー内の次の分岐またはリーフに対するポインタを含むプログラム命令と、非空のテーブル・エントリ内の前記ポインタが、前記対応する検索ツリーのリーフまたは次の分岐のどちらを指すかを判定するプログラム命令と、前記ポインタが、前記対応する検索ツリーの前記リーフを指さない場合に、前記次の分岐内容を読み込むプログラム命令と、対応する検索ツリーの前記リーフに到達したときに前記リーフ内容を読み込み、前記リーフ内的一对のパターンと前記検索キーを比較して前記一对のリーフ・パターンによって定義された範囲が前記検索キーを含むかどうかを判定するプログラム命令と、前記リーフ・パターンが前記検索キーを含む場合に、見つかった前記リーフの内容を前記要求するアプリケーションに戻すプログラム命令とを含むコンピュータ可読媒体。

(37) プログラム可能ハッシュ関数を用いて前記入力キーをハッシュして検索キーを形成するプログラム命令をさらに含む、上記(36)に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

(38) 検索ツリーの複数のルート・ノードを表す前記テーブルが、2^Nのエントリを含む、上記(36)に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

(39) 前記コンピュータ処理デバイスがネットワーク

・プロセッサである、上記(36)に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

(40) 前記対応する検索ツリーの前記次の分岐の内容が、他の次の分岐を指す、上記(36)に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

(41) 前記次の分岐の内容が、前記対応する検索ツリーの前記リーフを指す、上記(36)に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。 10

(42) 前記リーフ・パターンが、前記検索キーを含まず、他のリーフに対するポインタを含まない場合に、成功表示を戻さないプログラム命令をさらに含む、上記

(36)に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

(43) 前記テーブルに対する前記インデックスが空のエントリを指している場合に、成功表示を戻さないプログラム命令をさらに含む、上記(36)に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。 20

(44) カラー・レジスタの内容を前記検索キーに追加して最終検索キーを提供するプログラム命令をさらに含む、上記(36)に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

(45) 前記検索キーに0の文字列を追加して最終検索キーを提供するプログラム命令をさらに含む、上記(36)に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

(46) 前記検索キー内の前記諸ビットが、前記一对のパターンによって定義される範囲内にあるかどうかを判定するためにチェックされる整数として扱われる、範囲下の比較動作を実行するプログラム命令をさらに含む、上記(36)に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。 30

(47) 前記検索キー内の諸ビットが、第1リーフ位置の諸ビットと、第2リーフ・パターンに指定されたマスク下で比較される、マスク下の比較動作を実行するプログラム命令をさらに含む、上記(36)に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。 40

(48) 前記リーフが他のリーフに対する連鎖ポインタを含む場合に、他のリーフに記憶された一对のパターンを読み込み、前記パターンと前記検索キーを比較するプログラム命令と、記憶された前記パターンが、前記検索キーを含まず、前記連鎖内の次のリーフに対するポインタを含まない場合に、成功表示を戻さないプログラム命令とをさらに含む、上記(36)に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

(49) 前記リーフが他のリーフに対する連鎖ポインタを含む場合に、他のリーフに記憶された一对のパターンを読み込み、前記パターンと前記検索キーを比較するプログラム命令と、記憶された前記パターンが前記検索キーを含む場合に、成功表示を戻すプログラム命令とをさらに含む、上記(36)に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

【図面の簡単な説明】

【図1】本発明の好ましい実施形態による、ネットワーク・プロセッサのための一例示的アーキテクチャを示す図である。

【図2】本発明の好ましい実施形態による、組込みプロセッサ複合体のための一例示の実施形態を示す図である。

【図3】本発明の好ましい実施形態による、一例示的プロトコル・プロセッサ構造を示す図である。

【図4】本発明の好ましい実施形態による、一例示的流入／流出フレームの流れを示す図である。

【図5】本発明の好ましい実施形態による、完全一致検索アルゴリズムのためのツリー・データ構造を示す図である。

【図6】本発明の好ましい実施形態による、一例示的データ構造に対する直接テーブルを使用することの効果を示す図である。

【図7】本発明の好ましい実施形態による、一例示的データ構造に対する、直接リーフをイネーブルにすることの効果を示す図である。

【図8】本発明の好ましい実施形態による、ソフトウェア管理ツリー(SMT)のための入力キーおよびリーフ・パターン中のフィールドの例を示す図である。

【図9】本発明の好ましい実施形態による、比較定義テーブル・エントリのための一例示的フォーマットを示す図である。

【図10】本発明の好ましい実施形態による、ソフトウェア管理ツリー検索アルゴリズムの処理ロジックを示す図である。

【図11】本発明の好ましい実施形態による、一例示的ルックアップ定義テーブルの内部構造を示す図である。

【図12】PSCBレジスタの内部フォーマットを示す図である。

【図13】SMTツリーのための固定リーフ・フォーマットを示す図である。

【図14】本発明の好ましい実施形態による、ツリー検索エンジンのための一例示的アーキテクチャを示す図である。

【符号の説明】

10 ネットワーク・プロセッサ

12 組込みプロセッサ複合体

14 エンキュー・デキュー・スケジューリング論理回

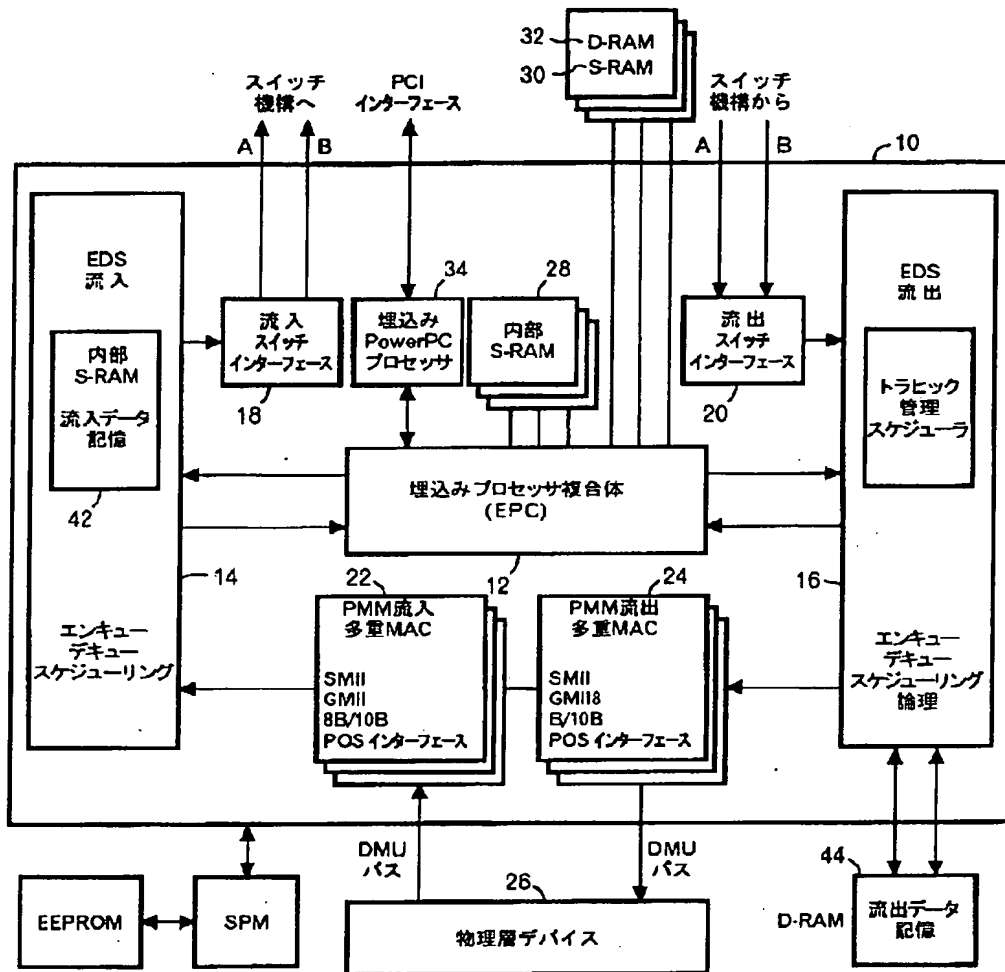
35

36

16 エンキュー・デキュー・スケジューリング論理回路
 18 流入スイッチ・インタフェース
 20 流出スイッチ・インタフェース
 22 物理MACマルチプレクサ
 24 物理MACマルチプレクサ
 26 物理層デバイス

28 内部SRAM
 30 外部ZBT SRAM
 32 外部DDR DRAM
 34 制御ポイント・プロセッサ
 42 流入データ記憶
 44 流出データ記憶

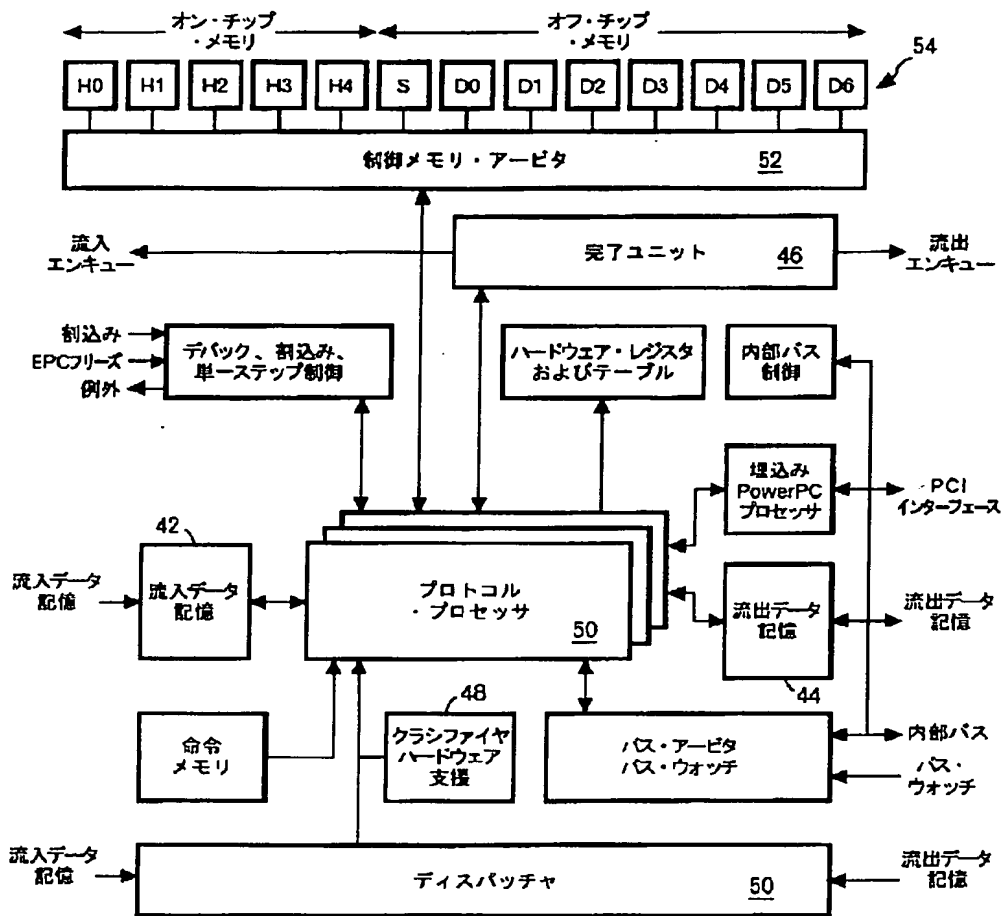
【図1】



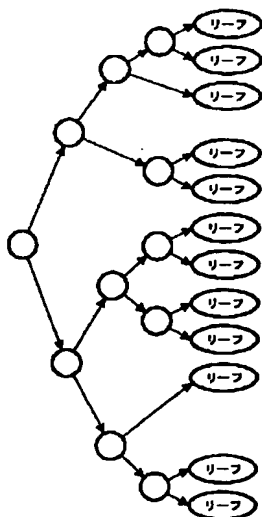
【図8】

入力キー	IPSA (32 bits)	IPDA (32 bits)	ソース・ポート (16 bits)	宛先ポート (16 bits)	プロトコル (8 b)
リーフ パターン 0	値	値	Min	Min	値
リーフ パターン 1	マスク	マスク	Max	Max	マスク

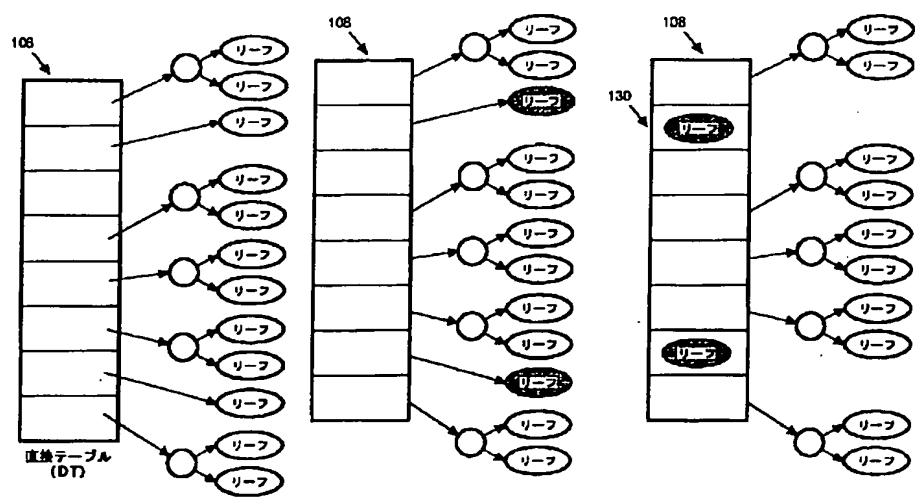
【図2】



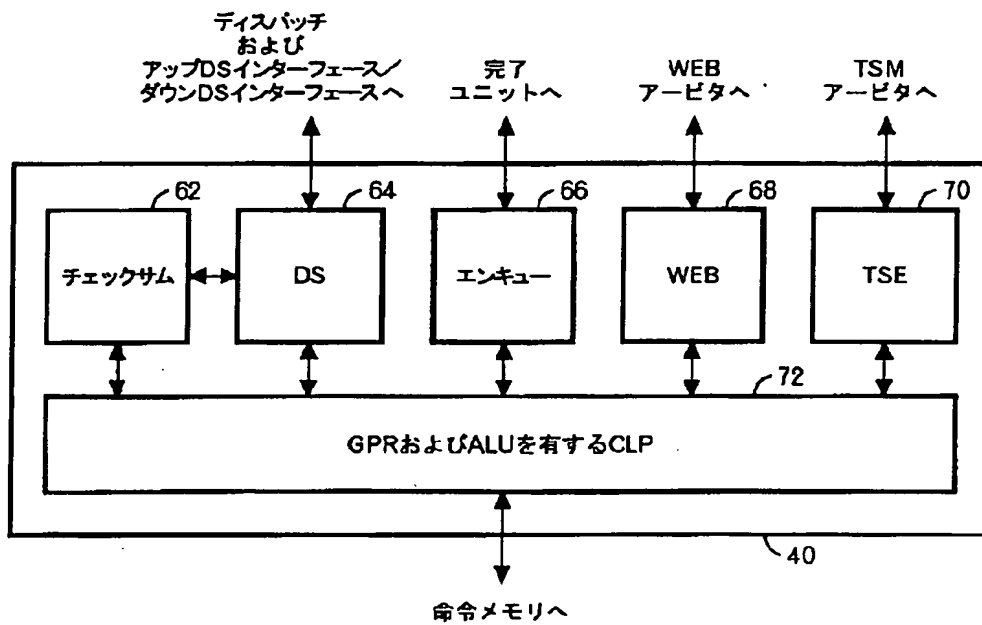
【図6】

直接テーブルを使用しない
データ構造

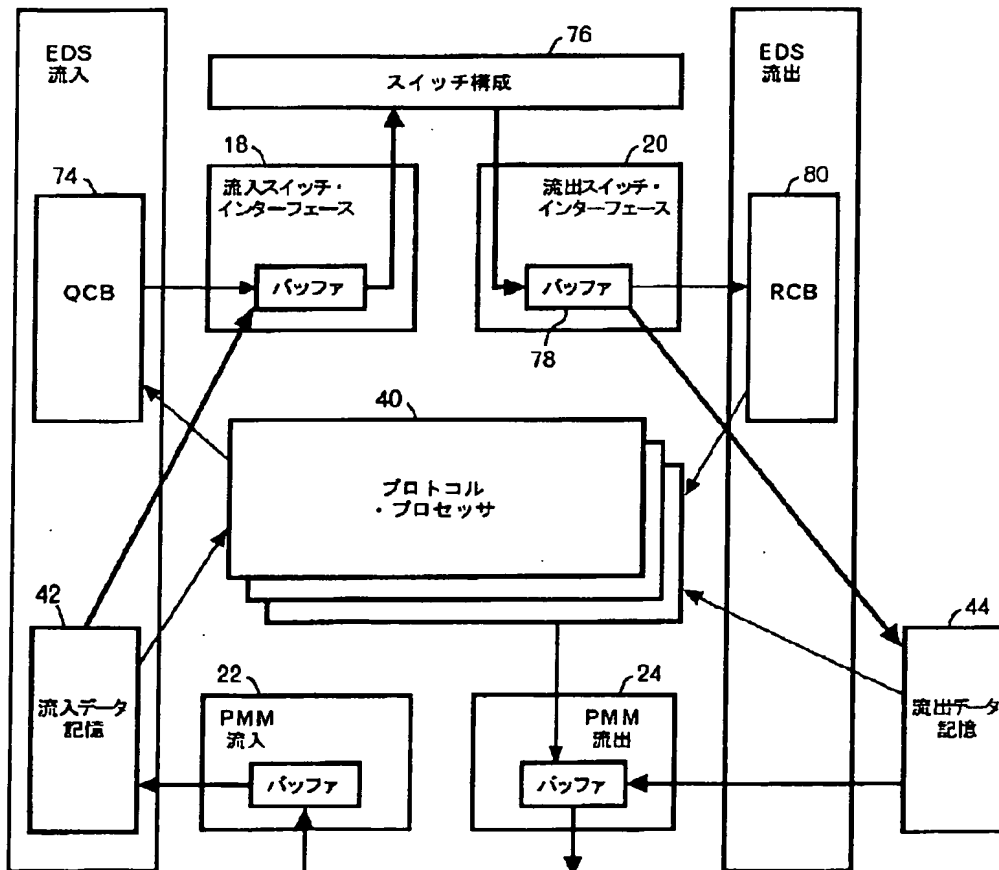
【図7】

直接テーブルを使用する
データ構造直接リーフのない
データ構造直接リーフのある
データ構造

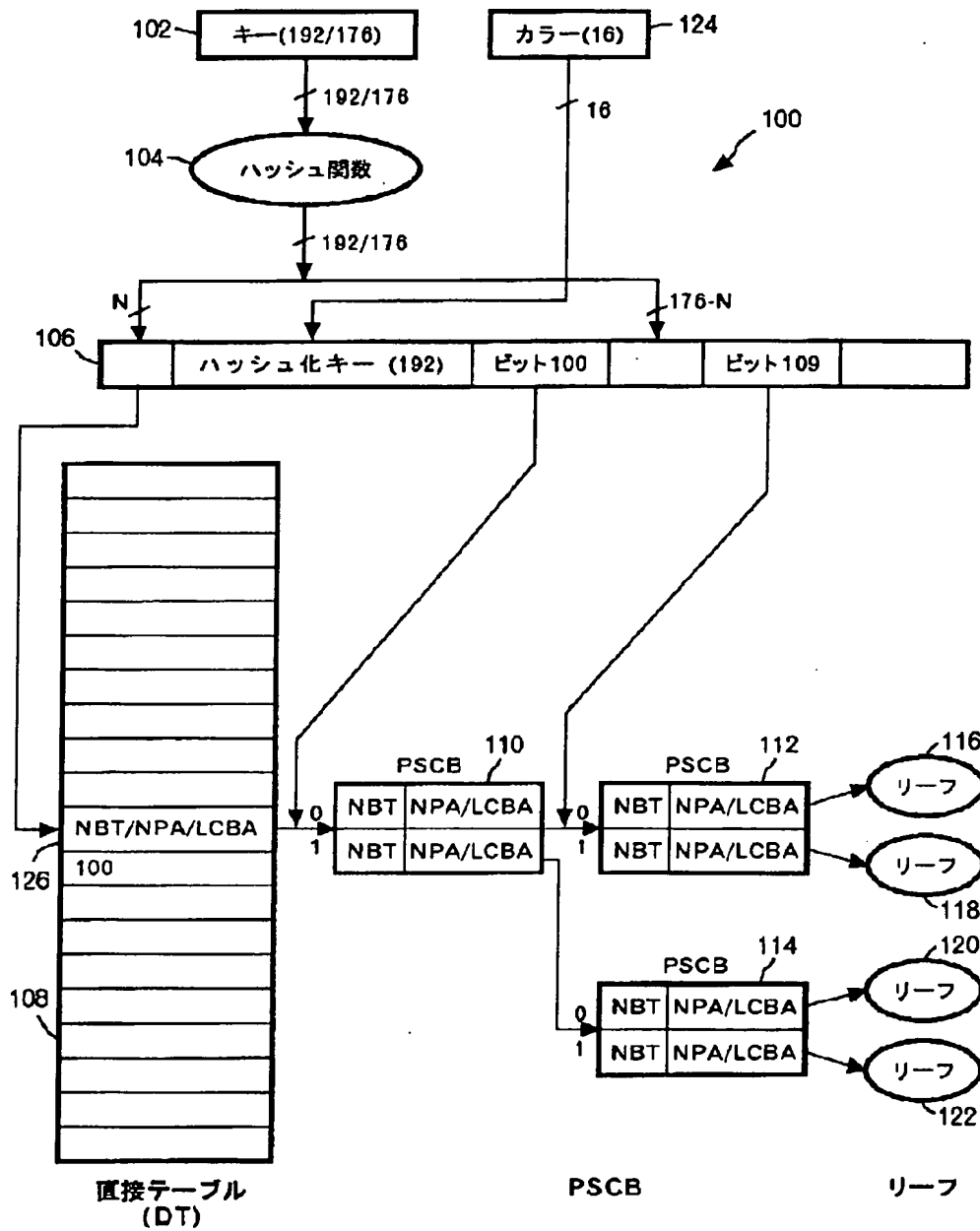
【図3】



【図4】



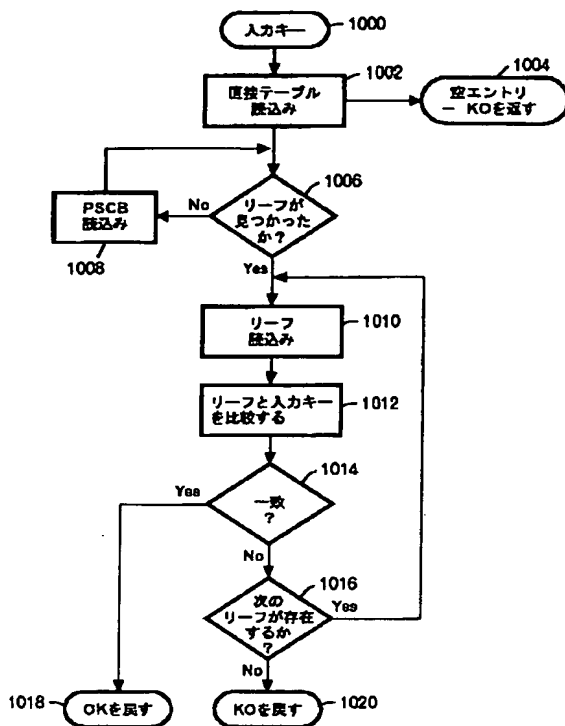
【図5】



【図9】

範囲 1		範囲 2			
オフセット フィールド0	最小最大 長さ フィールド0	オフセット フィールド1	最小最大 長さ フィールド1	1 範囲 / 2 範囲	継続
8 bits	8 b	8 bits	8 b	1 bit	1 bit
				0 : 範囲 1 が妥当 1 : 範囲 1、2 が妥当	0 : この比較で停止 1 : 次の比較で継続

【図10】



【図11】

LUDefTableツリー定義		
フィールド	サイズ	ビット
CacheEntry	1	0
Tree Type	2	2..1
hash-type	4	6..3
color_en	1	7
PIP2_max_size	5	12..8
NPARops_en	1	13
NPASMT_en	1	14
Complindex_en	1	15
PSCB_fg_index	6	21..16
PSCB_Height	1	22
Mask_Vector_En	1	23
Complindex	8	31..24
DT_base_addr	26	57..32
DT_size	4	61..58
DT_interleaf	2	63..62
Leaf_fg_index	6	69..64
Leaf_Width	2	71..70
Leaf_Height	3	74..72
DirectLeafEn	1	75

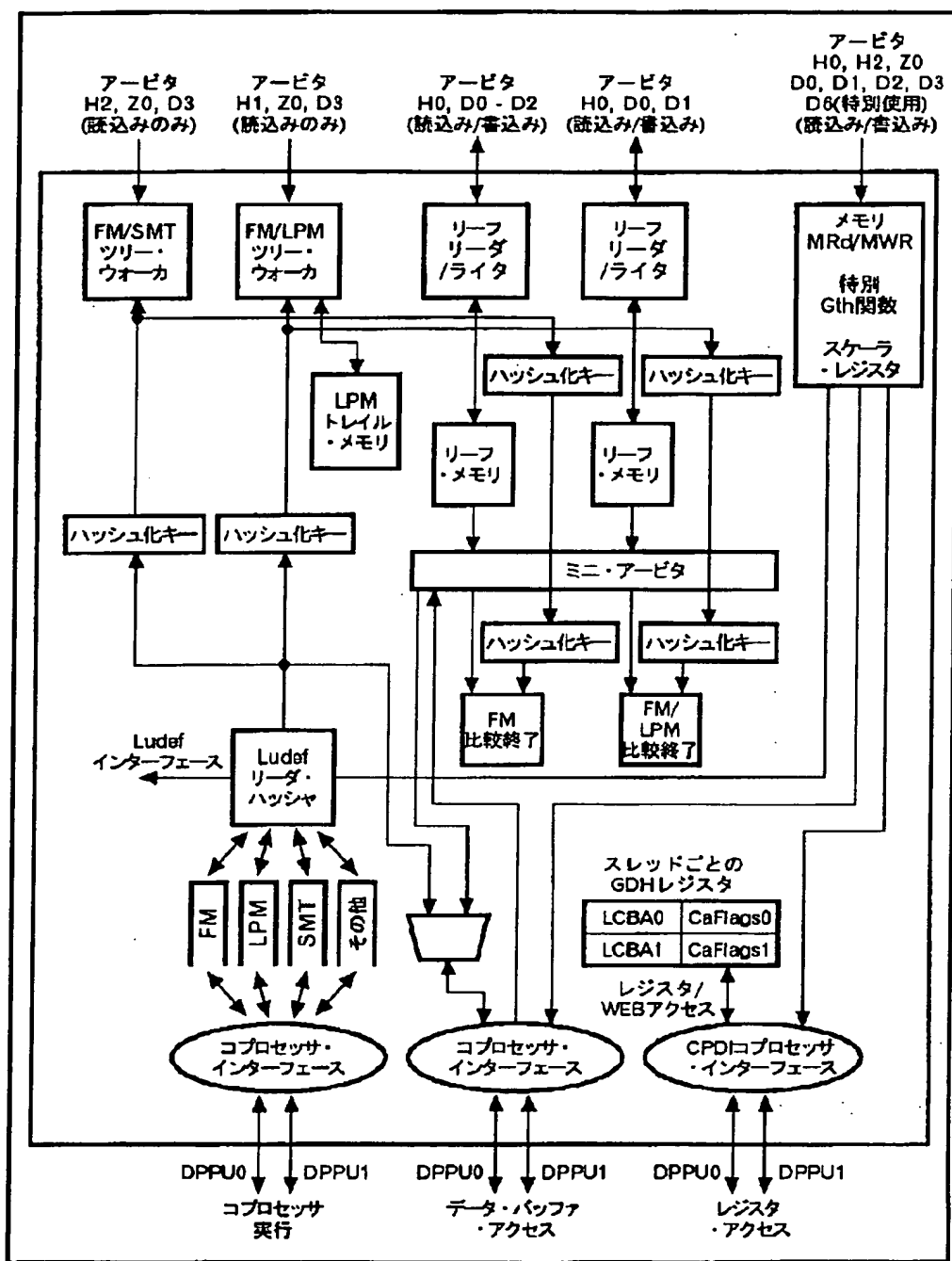
【図12】

フィールド	サイズ	PSCBが配置されるTSM内のアドレス
NPA0	26	次のPSCBアドレス： PSCBの0-部分のための、ツリー内の次のPSCBに対するポインタ
NBT0	8	PSCBの0-部分のための、テストすべき次のビット
NCBA0	26	リーフ制御ブロック・アドレス： PSCBの0-部分のためのリーフに対するポインタ
NPA1	26	次のPSCBアドレス： PSCBの1-部分のための、ツリー内の次のPSCBに対するポインタ
NBT1	8	PSCBの1-部分のための、テストすべき次のビット
LCBA1	26	リーフ制御ブロック・アドレス： PSCBの1-部分のためのリーフに対するポインタ
Index	8	このPSCBのインデックス (前のPSCBに物理的に記憶されている)
PatBit	1	PSCBレジスタ内のインデックス・フィールドの 値に基づく Hashedkey[インデックス]の値

【図13】

フィールド名	長 さ	内 容
NLASMT	4 バイト	SMTのリーフを結びつけるためのリーフ連鎖ポインタ 結びつけられたリーフの形を含む
Comp_table_index	1 バイト	Pattern1、Pattern2とHashedKeyの間の比較を定義する CompDefTable内のインデックスを定義する
Pattern1 and Pattern2	4-36 バイト	このフィールドは、Pattern1およびPattern2を ビットごとに交互に含む すなわちこのフィールドのビット0はPattern1の ビット0を含み、ビット1はPattern2のビット0を 含む 交互に偶数ビットはPattern1を表し、奇数ビットは Pattern2を表す
UserData	可 変	このフィールドの内容は完全なビココード制御下にある； UserDataフィールドは1つまたは複数のカウンタを 含むことができる

【図14】



フロントページの続き

(72)発明者 ジャン・ルイ・カルヴィニャク
 アメリカ合衆国27511 ノースカロライナ
 州ケアリー スプリング・ホロー・レーン
 112

(72)発明者 マーコ・シー・ヘデス
 アメリカ合衆国27612 ノースカロライナ
 州ローリー グランド・マナー・コート
 4109 ナンバー308

(72)発明者 クラーク・デブス・ジャフリーズ
アメリカ合衆国27713 ノースカロライナ
州ダラム ベインブリッジ・ドライブ
2806エイチ
(72)発明者 ビュシ・チュニラル・パテル
アメリカ合衆国27513 ノースカロライナ
州ケアリー スモークモント・ドライブ
105

(72)発明者 マーク・アンソニー・リナルディ
アメリカ合衆国27713 ノースカロライナ
州ダラム キーンズベリー・サークル
1201
F ターム (参考) 5B075 KK02 ND02 ND34 ND35 NK13
NK24 NK43 NK45 PP22 QM01
QS13 UU16
SK030 KA01 KA02